

2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

AD-A267 314



DTIC
ELECTE
JUL 28 1993
S B D

THESIS

TARGET DETECTION VIA KALMAN FILTERING

by

Gary Alan Sipe

March 1993

Thesis Advisor:

Harold A. Titus

Approved for public release; distribution is unlimited.

93 7 28 06 4

93-16964

REPORT DOCUMENTATION PAGE			
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE		4. PERFORMING ORGANIZATION REPORT NUMBER(S)	
5. MONITORING ORGANIZATION REPORT NUMBER(S)		6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	
6b. OFFICE SYMBOL (If applicable) EC		7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION		8b. OFFICE SYMBOL (If applicable)	
8c. ADDRESS (City, State, and ZIP Code)		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
10. SOURCE OF FUNDING NUMBERS		11. TITLE (Include Security Classification) TARGET DETECTION VIA KALMAN FILTERING.	
12. PERSONAL AUTHOR(S) Gary Alan Sipe		13a. TYPE OF REPORT Master's Thesis	
13b. TIME COVERED From To		14. DATE OF REPORT (year, month, day) March 1993	
15. PAGE COUNT 84		16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.	
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUBGROUP	Kalman Filtering, Angle of Arrival, Time Difference of Arrival, Optimal Estimation
19. ABSTRACT (Continue on reverse if necessary and identify by block number)			
<p>A simple, time domain method is used to analyze moderate to high PRF radar signals. The quantities of interest are the signal's PRF, SNR, and time of arrival. The time of arrival problem is important because it can be used, with multiple sensors, to determine the position of the emitting target. An algorithm is described which will produce these values using Kalman filtering. Individual pulses in a pulsed type radar are measured against a threshold using a two sample detection scheme to provide some glitch rejection. Results of individual, time domain measurements of the signal parameter are smoothed with a Kalman filter. Integrating the pulse train envelope during the radar dwell time provides the energy centroid for a scan cycle. This centroid, time differenced with multiple sensors provides observables for an Extended Kalman Filter for emitter localization. The work here simulates all data. Tests of the algorithms developed were conducted on real, classified data in addition to the work presented here.</p>			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS REPORT <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Harold A. Tikus		22b. TELEPHONE (Include Area Code) (408)-656-2560	
22c. OFFICE SYMBOL EC/Ts			

Approved for public release; distribution is unlimited.

Target Detection Via
Kalman Filtering

by

Gary A. Sipe
Lieutenant, United States Navy
B.S. in Chemical Engineering, Tri State University
B.S. in Electrical Engineering, Naval Postgraduate School, Monterey

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

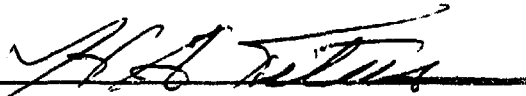
NAVAL POSTGRADUATE SCHOOL
March 1993

Author:

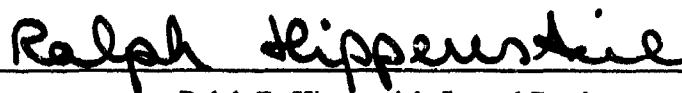


Gary A. Sipe


Approved by:



Harold A. Titus, Thesis Advisor



Ralph D. Hippenstiel, Second Reader



Michael A. Motgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

A simple, time domain method is needed for analyzing moderate to high PRF radar signals. The quantities of interest are the signal's PRF, SNR, and time of arrival. The time of arrival problem is important because it can be used, along with information from multiple sensors, to determine the position of the emitting target. An algorithm is described which will produce these values using optimal estimation via Kalman filtering. Individual pulses in a pulsed type radar are measured against a threshold using a two sample detection scheme to provide some glitch rejection. Results of individual, time domain measurements of the signal parameters are smoothed with a Kalman filter. Integrating the pulse train envelope during the radar dwell time provides the energy centroid for a scan cycle. The work here simulates all data and processing in MATLAB. Tests of the algorithms developed were conducted on real, classified data in addition to the work presented here.

DTIC QUALITY INSPECTED 5

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

TABLE OF CONTENTS

I. INTRODUCTION	1
A. ELECTRONIC INTELLIGENCE	1
B. GOALS FOR THIS RESEARCH	2
II. BACKGROUND INFORMATION	3
A. BASIC RADAR PARAMETERS	3
1. Scan Rate/Scan Pattern	3
2. PRI/PRF Analysis	6
3. Pulse Width/Pulse Shape Analysis	7
B. THE TDOA GEOMETRY	7
1. Hyperbolic Lines of Position	8
2. The Angle of Arrival Approximation	10
III. RADAR DATA SETS AND THEIR SPECTRA	13
A. OBTAINING THE DIGITAL DATA	13
B. TYPICAL SIGNALS OF INTEREST	14

IV. TIME DOMAIN BASED SIGNAL ANALYSIS	21
A. PRF/PRI MEASUREMENT	21
B. SCAN RATE MEASUREMENT	25
C. TIME OF ARRIVAL	27
D. SNR AND THRESHOLD DETERMINATION	28
V. PERFORMANCE RUNS AND RESULTS	30
A. EMITTER WITH ZERO NOISE	31
B. EMITTER WITH ADDED NOISE	37
VI. SUMMARY	44
A. CONCLUSIONS	44
B. RECOMMENDATIONS	44
APPENDIX A. ELINT RANGE ADVANTAGE	46
APPENDIX B. MATLAB SOURCE CODE	50
LIST OF REFERENCES	74
INITIAL DISTRIBUTION LIST	75

LIST OF FIGURES

Figure 1.	Typical Configuration of a Long Range Search Radar	5
Figure 2.	The Hyperbolic Line of Position Geometry	9
Figure 3.	The Angle of Arrival Geometry	12
Figure 4.	Signal to Data Conversion and Terminology	15
Figure 5.	Pulse Burst (Sample Rate = 1 MHz)	17
Figure 6.	Pulse Burst Spectrum (Sample Rate = 1 MHz)	18
Figure 7.	Pulse Burst (Sample Rate = 400 kHz)	19
Figure 8.	Pulse Burst Spectrum (Sample Rate = 400 kHz)	20
Figure 9.	Emitter with Zero Noise	32
Figure 10.	PRF Histories for the Zero Noise Emitter	33
Figure 11.	AOA Estimates for Zero Noise Emitter	34
Figure 12.	Closeup of AOA Estimates for Zero Noise Target	35
Figure 13.	Emitter With Gaussian Noise	39
Figure 14.	PRF History for Trial #1	40
Figure 15.	PRF History for Trial #2	41
Figure 16.	PRF History for Trial #3	42
Figure 17.	AOA Estimates for Noisy Emitter	43
Figure 18.	ELINT Range Advantage	49

ACKNOWLEDGMENTS

I would like to thank Professor Harold Titus for his insight, patience, and guidance as my thesis advisor. I would also like to give special thanks to Professor Ralph Hippenstiel for his superlative instruction during the preparation of this work. His hours of willing assistance were invaluable to this research. To my wonderful wife, Clare, and my son, Eric, thank you for the unconditional love and support you have shown over the past two years. I dedicate this thesis to both of you.

I. INTRODUCTION

A. ELECTRONIC INTELLIGENCE

Radar has been called the greatest advance in the sensing of remote objects since the invention of the telescope in 1608 [Ref. 1]. Despite the numerous advantages of using radar for locating and classifying remote objects, there is a serious flaw. A large amount of energy must be transmitted in order to produce a radar return suitable for processing. Electronic Intelligence (ELINT) is the process of observing this energy passively and using it for locating or classifying a remote, radio frequency (RF) emitting object.

Because a radar signal must travel twice the distance from the object to the radar transmitter, ELINT processing should carry a range advantage of at least two. Strict numerical treatment of the range advantage of ELINT over radar is given in Appendix A. ELINT also has the advantage of not requiring any energy emission to accomplish its task. This allows the ELINT receiver to remain covert.

It is relatively easy to intercept a radar signal (they typically are of a high power density). If the signal can be detected, then comparison with another receiver will give the angle of arrival (AOA) of the signal relative to the sensor pair. This is useful for establishing the position of an emitter if it is stationary. If the emitter is moving, further processing of the angles of arrival can be performed to estimate the emitter track.

Despite current trends in the field of Low Probability-of-Intercept (LPI) radars, there can be no doubt that ELINT will still be used. The concept of an LPI radar has been

described as trying to illuminate an incoming aircraft with a search light without having the aircrew able to see the spotlight [Ref. 2]. Though LPI radars may require special techniques and procedures to be detected, it is unlikely that they will remain immune to detection for long.

B. GOALS FOR THIS RESEARCH

A simple, time domain algorithm which may be used to process radar data from a dual sensor array is desired. The method must extract useful times of signal arrival in order to provide a tracking algorithm with angles of arrival of the signal. The method must also produce good estimates of primary radar signal parameters. Of specific interest for this work are the radar pulse repetition frequency (PRF), the scan rate, and the signal-to-noise ratio at the receiver (SNR). The precise meaning of the terms PRF and scan rate, as they apply to this work, are discussed in the next chapter. Some *a priori* knowledge of the radar pulse width is assumed.

The method advocated here requires a minimum amount of data to be passed between the receiver platforms. This lowers the required bandwidth of the communications channel between the receivers and also eliminates the need for a central processing station on the ground. Times of arrival are generated by both platforms and synchronized against a highly accurate master clock. Times of arrival, synchronization information, and receiver position information are all that must be transferred.

II. BACKGROUND INFORMATION

A. BASIC RADAR PARAMETERS

There are numerous parameters which can be used to describe the characteristics of a given radar system. In this section, we will review those of interest for this work. For further information regarding radar parameters of interest to ELINT processing consult *Electronic Intelligence: The Analysis of Radar Signals* by Richard G. Wiley [Ref. 1].

1. Scan Rate/Scan Pattern

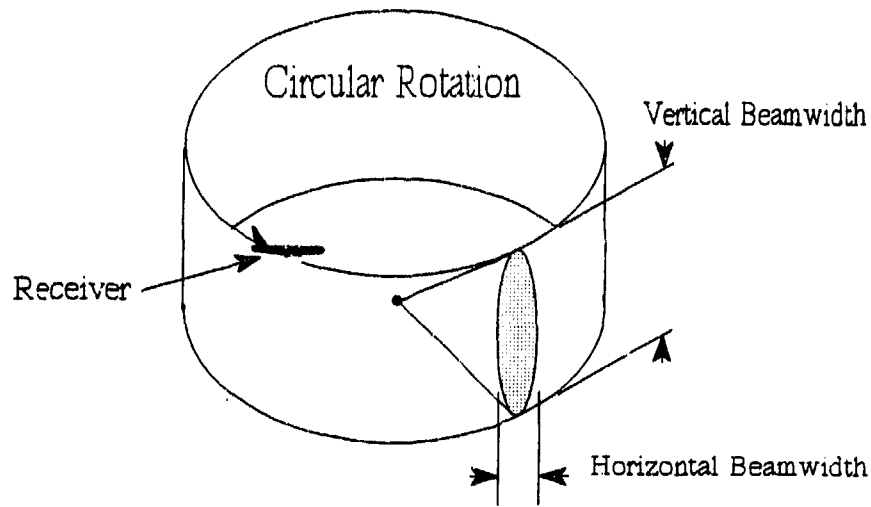
The beam from a radar antenna typically covers only a small portion of the angular area of interest. Therefore, the beam is moved over time. Many different patterns may be used to sweep out a volume. The scanning pattern is designed to cover the volume of interest of the radar. The size of the volume of interest may change over time. It is not unusual for a radar sensor to search through a hemisphere in order to acquire targets. Once the target has been acquired, smaller volumes will be analyzed to perform the function of tracking. The change in scan rate or pattern associated with the switch from search mode to track mode or to a separate tracking radar is of interest.

Another use of scan pattern information is in determining possible capabilities of the radar emitter. Radars with mechanically produced scan patterns tend to be periodic. Electronically switched scan patterns may or may not be periodic and typically can change scan patterns rapidly. Radars with more complicated scan patterns tend to have more capability for target localization and tracking than those with simple ones.

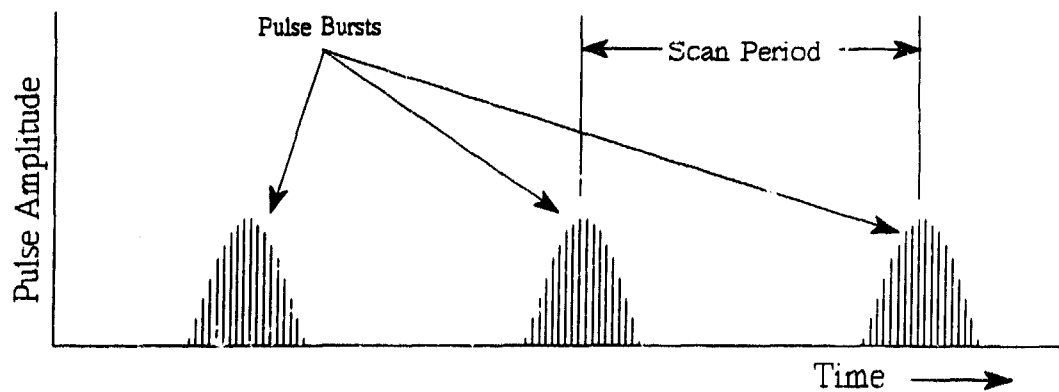
There are fan beam and pencil beam radars. Pencil beam radars have very narrow antenna beams and use complex scan patterns to cover their volumes of interest. Typical search patterns for pencil beam radars are raster, helical, and spiral scans.

The second classification, fan beams, have radar antenna beams which are characteristically much wider in one direction than another. This means that the radar has very broad coverage in one direction, but poor resolution. Therefore, the radar is moved perpendicularly to the plane along the broadest coverage line. A common configuration for a long-range, early warning type radar is to have a fan beam radar which scans in azimuth. Figure 1 after [Ref. 1] shows the geometry involved as well as the signature seen at the ELINT receiver. This arrangement gives good long range detection capacity with high accuracy in azimuth, but poor elevation information. A second height-finding radar can be used to gather elevation information [Ref 1]. This work will concentrate on detecting signals from these types of radar installations. These often provide long range detection or early warning capability and are of interest. For the remainder of this work, we will be interested solely in these types of acquisition radars. The term PRF will be used to mean the frequency at which pulses are emitted from the radar. The term scan rate will apply to the number of revolutions in azimuth that the radar makes per unit time.

The signal bursts in Figure 1 are free of background noise, but show the essential pattern to be sought in sampled data from an ELINT receiver. Here we present a possible method for estimating the energy centroid of such signal bursts in order to provide accurate times of pulse burst arrivals. Correlating information from two sensors about these times of arrival will give the direction of the signal (DOA). An algorithm is



(a) CIRCULAR SCAN REGION



(b) RECEIVED AMPLITUDE VERSUS TIME

Figure 1. Typical Configuration of a Long Range Search Radar

proposed to obtain accurate centroids simply using a decision tree for detecting individual pulses inside the burst and then smoothing them with a Kalman Filter.

The other commonly used pattern for a fan beam radar is sector scanning. This is used when only a portion of the azimuth circle is required for coverage. Height finding radars generally use a sector scan in elevation when coupled with early warning radars. Sector scan radars produce pulse bursts identical to those of circular scan radars. However, the bursts will not necessarily come at even intervals. This work will not attempt to identify sector scan patterns.

2. PRI/PRF Analysis

The pulse repetition interval (PRI) is an important radar parameter to evaluate. For a radar with a fixed PRI, the maximum unambiguous range and velocity are given by the following equations:

$$R_u = \frac{c \times PRI}{2}; \quad (1)$$

and

$$V_u = \frac{c}{(2 \times PRI \times RF)}; \quad (2)$$

where

R_u = Maximum unambiguous range (m),
 V_u = Maximum unambiguous radial velocity (m/s),
PRI = Pulse Repetition Interval (s),
RF = Radar carrier frequency (Hz), and
 c = Speed of light.

These two equations provide more than just information about the emitter platform capabilities. Information such as the maximum range of the radar could serve as the basis for initial estimates of emitter position. For high PRF radars, the unambiguous range is very short. Ranging, then, is performed by other techniques such as FM ranging (chirp). In this case, Equations (1) and (2) are not valid.

There are radars with constant PRI and jittered PRI. In this simulation only signals of constant PRI were used. The method should be able to estimate an average of the radar PRI for all types of radars, including those with PRI jitter. However, the resolution of the techniques in this work is not sufficient to accurately measure the magnitude of PRI jitter.

3. Pulse Width/Pulse Shape Analysis

Finally, we will briefly discuss the importance of pulse shape. For any digital sampling method to detect a pulse stream from a radar, the sampling frequency must be at a minimum of one sample is taken over the pulse width. If this is not true, then it is possible for pulses to go undetected. This will be seen to adversely affect time difference of arrival (TDOA) methods.

B. THE TDOA GEOMETRY

This section introduces the concepts and equations necessary to convert TDOA data into useful estimates of the angle of arrival of a signal.

1. Hyperbolic Lines of Position

For the TDOA problem, the essential parameter is the arrival time difference Δt of a signal between two (or more) receivers. The resulting time difference measured at the receivers could be the result of an emitter anywhere on a hyperbola constituting possible positions. This is called the hyperbolic line-of-position (LOP) of the emitter. The locus of points on which the emitter could lie for a given Δt is given in Figure 2. For this discussion, both receivers will always be assumed to lie on the x-axis. This assumption allows the hyperbola to be easily evaluated and can be obtained by a simple coordinate transformation.

A hyperbola is the set of points in a plane whose distances from two fixed points in the plane have a constant difference. The two fixed points are the foci of the hyperbola [Ref. 3]. For the TDOA problem, the foci of the hyperbola are the receivers. The hyperbola can be constructed knowing only the absolute value of the Δt . If we know the sign as well, i.e., which receiver was first to intercept the signal, then one side of the hyperbola may be ignored. Letting d be the distance between the receivers the following relationship is true:

$$\Delta t_{\max} = \frac{d}{c} ; \quad (3)$$

where

d = is the receiver separation distance,
 Δt_{\max} = the maximum possible Δt based on receiver separation, and
 c = the speed of light.

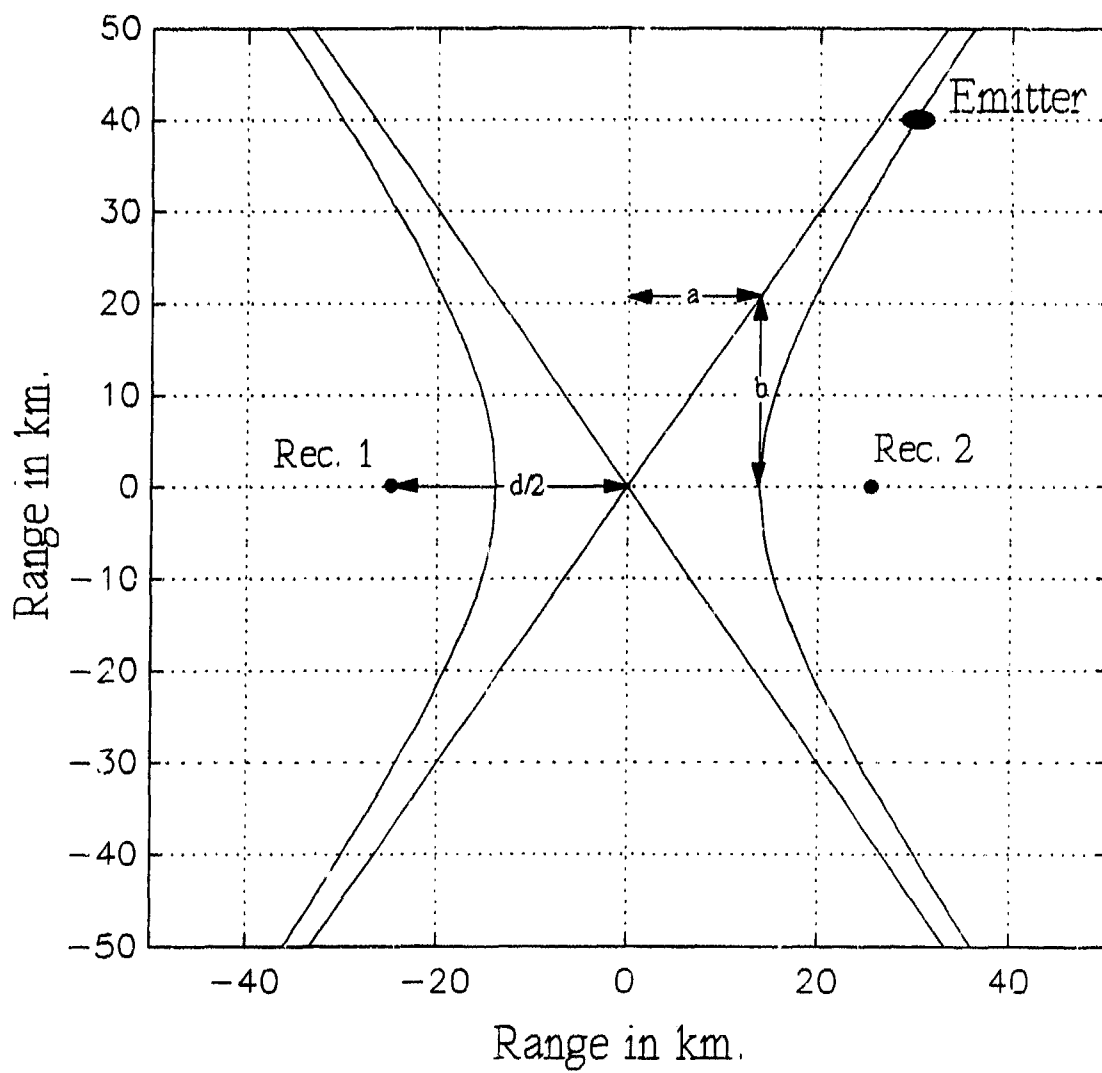


Figure 2. The Hyperbolic Line of Position Geometry

Once the Δt_{\max} is established, the line of position (LOP) calculation follows easily. The center to foci distance is simply $d/2$, so the center to intercept distance is

$$a = \frac{d}{2} \frac{\Delta t}{\Delta t_{\max}} . \quad (4)$$

The equation of any hyperbola is given by

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 ; \quad (5)$$

where

$$b = \sqrt{\left(\frac{d}{2}\right)^2 - a^2} . \quad (6)$$

Equation (5) is the hyperbola of position for the emitter for the TDOA problem. Figure 2 shows the quantities graphically. The hyperbolic line of position, as well as its asymptotes are shown. It is not evident from these equations, but if the sign of TDOA is known, then one curve of the hyperbola can be neglected. From Figure 2 it is clear that, at long range, we can approximate the emitter LOP just using the asymptotes to the hyperbola. The equation of the asymptotes is

$$y = \pm \frac{b}{a} x . \quad (7)$$

2. The Angle of Arrival Approximation

The usual ELINT situation involves long detection ranges relative to the receiver separation. Due to this, the correction for hyperbolic lines of position is not

necessary. We can use the approximate angle of arrival in order to describe the emitter location. Figure 3 shows the geometry after some simplifying assumptions. If the emitter is very distant, the approaching wave fronts of the radar signal are nearly straight, parallel lines. We can measure the Δt between receivers. Knowing the distance between receivers gives Δt_{\max} as before:

$$\alpha = \cos^{-1} \left(\frac{\Delta t}{\Delta t_{\max}} \right); \quad (8)$$

where

α = the angle of arrival (see Figure 3).

This approximation will be used for all LOP calculations. No treatment is given for the case of emitters which are too close for this approximation.

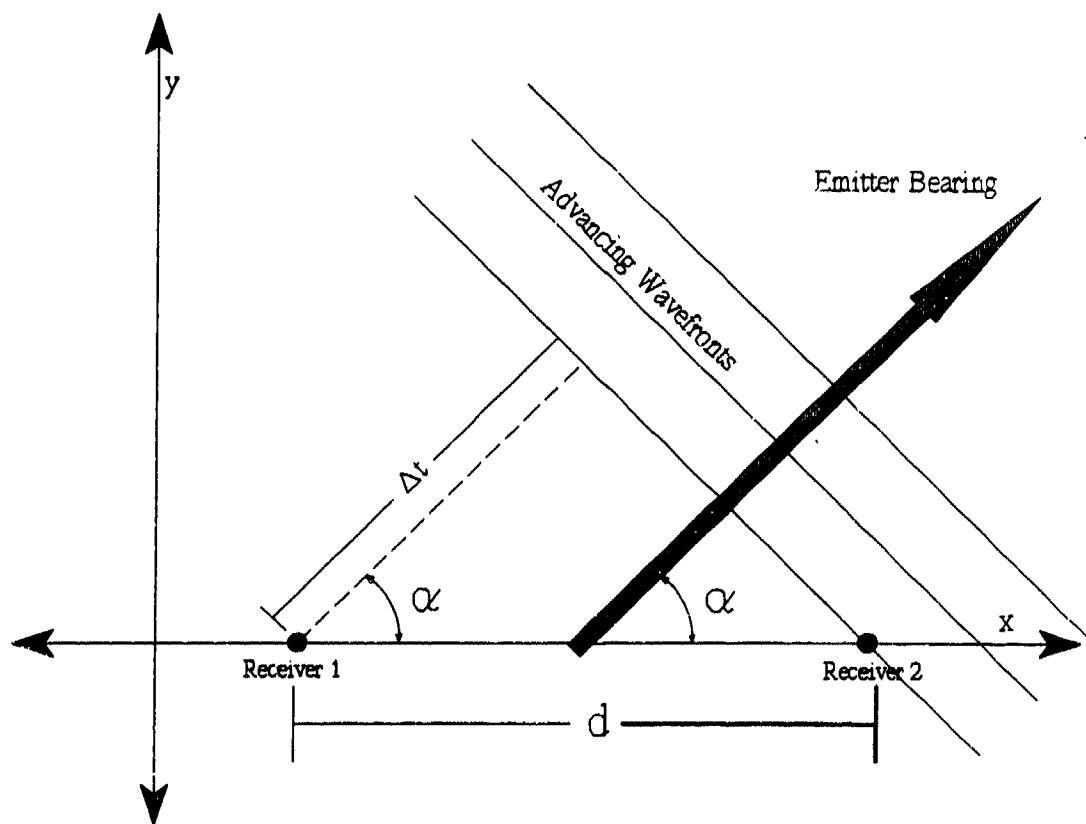


Figure 3. The Angle of Arrival Geometry

III. RADAR DATA SETS AND THEIR SPECTRA

A. OBTAINING THE DIGITAL DATA

In order to use a digital computer to process signal data, the data must be sampled to put it in a form which the computer can manipulate. This technology is readily available. The special consideration for radar signals is that they are of high frequency and, depending on which of the signal characteristics are to be examined, the sample rate may need to be very high. Measuring details about the individual pulses, for example, would require a sampling frequency greater than the Nyquist rate for the carrier frequency of the radar. This would severely tax the data handling capabilities of most computer systems. The method used here requires a sampling frequency which provides at least two samples per pulse. This turns out to be

$$SR \geq \frac{2}{pw}; \quad (9)$$

where

SR = the sample rate, and
pw = the pulse width of the radar emitter.

The radar signals of interest are from pulsed type radars. Continuous wave radars do not lend themselves to analysis by TDOA methods [Ref. 4]. The continuous wave radars will show a phase shift at two receivers at different positions. However, if the phase shift is greater than 2π , the number of phase shifts cannot be unambiguously

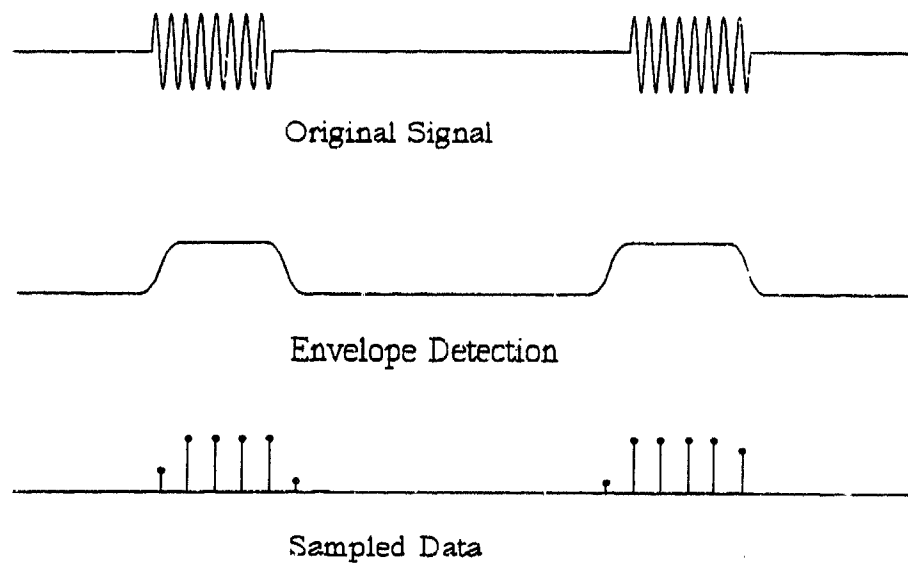
determined. It is possible for the same phenomenon to affect pulsed radars as well. One receiver could be far enough away that it is lagging by more than a single scan cycle. This is not likely since typical scan rates are on the order of 1 cycle per second or less. For one receiver to lag another by an entire scan cycle, assuming a rate of 1 scan per second, would require a separation distance of 300,000 kilometers.

The typical shape of a pulsed radar with no frequency modulation is seen in top portion of Figure 4a. The signal must be envelope detected so that digital sampling can occur. The frequency of the carrier wave is in the GHz range and cannot be sampled at a Nyquist rate, hence the envelope detection method must be used. By this method we now have approximately a square wave with a fixed duty cycle. This can easily be converted to digital data as illustrated in Figure 4a.

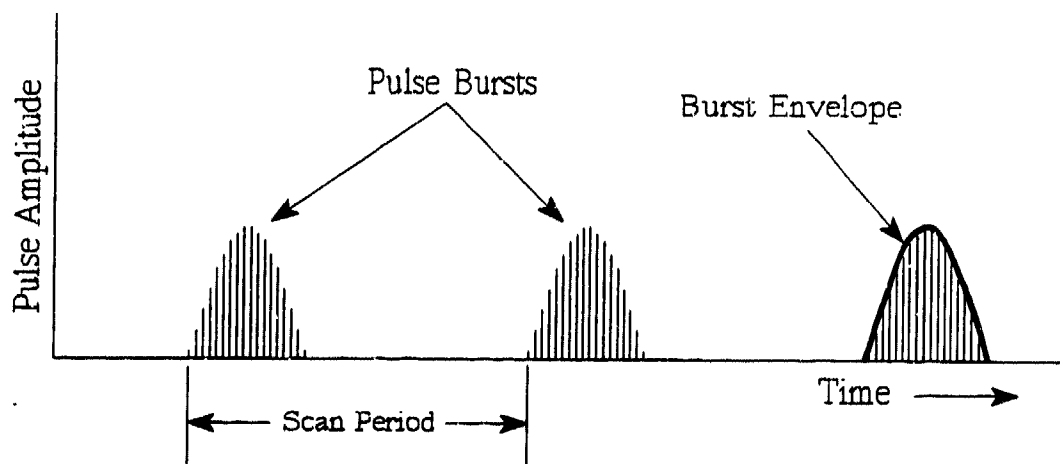
The amplitude of the square wave increases as the radar increases in strength during the scan. Peak occurs at the boresight of the radar. One half period of a sine function was used in constructing simulated data sets. Figure 4b provides another view of a data set with the characteristic features labelled.

B. TYPICAL SIGNALS OF INTEREST

The previous section describes the basic procedure for obtaining the digital data used. Here we will examine the data more closely. All emitters are simulated using the MATLAB programming language. Parameters such as PRF and scan rate were chosen by the author to be convenient numbers. Figure 5 shows one burst from a simulated data



(a) Transition to Sampled Data



(b) Signal Terminology

Figure 4. Signal to Data Conversion and Terminology

set used in this research. This data simulated a radar with a PRF of 9.0 kHz and a pulse width of 5 μ s. The pulse burst is shown without the presence of noise. The top graph shows the entire pulse burst. A cosine function is used to produce the burst envelope (the change in amplitude of the received signal as the main beam sweeps past the receiver). The second graph shows a closeup of some of the pulses inside the burst.

We are interested in the frequency components of the typical pulse burst so we will examine its frequency spectrum. The power spectral density was computed and is displayed in the upper graph of Figure 6. From the first graph we can see that the first null in the spectrum occurs at 200 kHz which corresponds to $1/(\text{pulse width})$. In the lower graph we observe that the first spectrum line occurs at the pulse repetition frequency of 9.0 kHz. These two parameters are readily evaluated from the spectrum of the data.

Figures 7 and 8 are included to show the effect of sampling rate on the pulse burst and the raw data set. The plots are produced in exactly the same method as before, but at a sample rate of 400 kHz. In the previous example, the sampling rate was 1 MHz. As we can see in Figure 7, there are only two samples inside each radar pulse. In Figure 8 the spectrum shows us that the null occurs at the very edge of the spectrum and cannot be used to estimate the pulse width. The methods discussed in the following sections of this chapter assume a sampling rate at or very close to twice the pulse width. This ensures good detection performance with a minimum of data. This also accounts for the statement in the introduction that *a priori* knowledge of the pulse width is required.

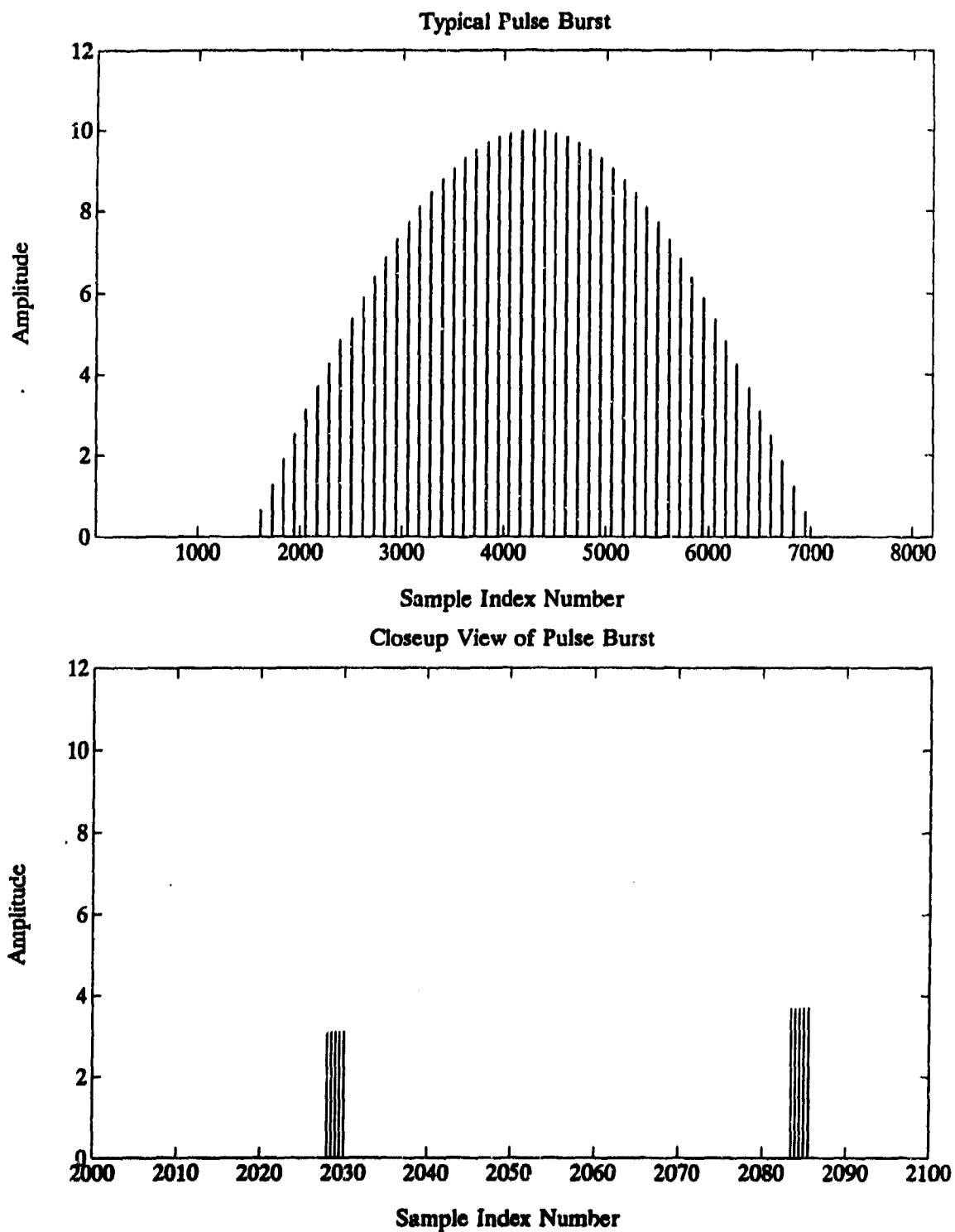


Figure 5. Pulse Burst (Sample Rate = 1 MHz)

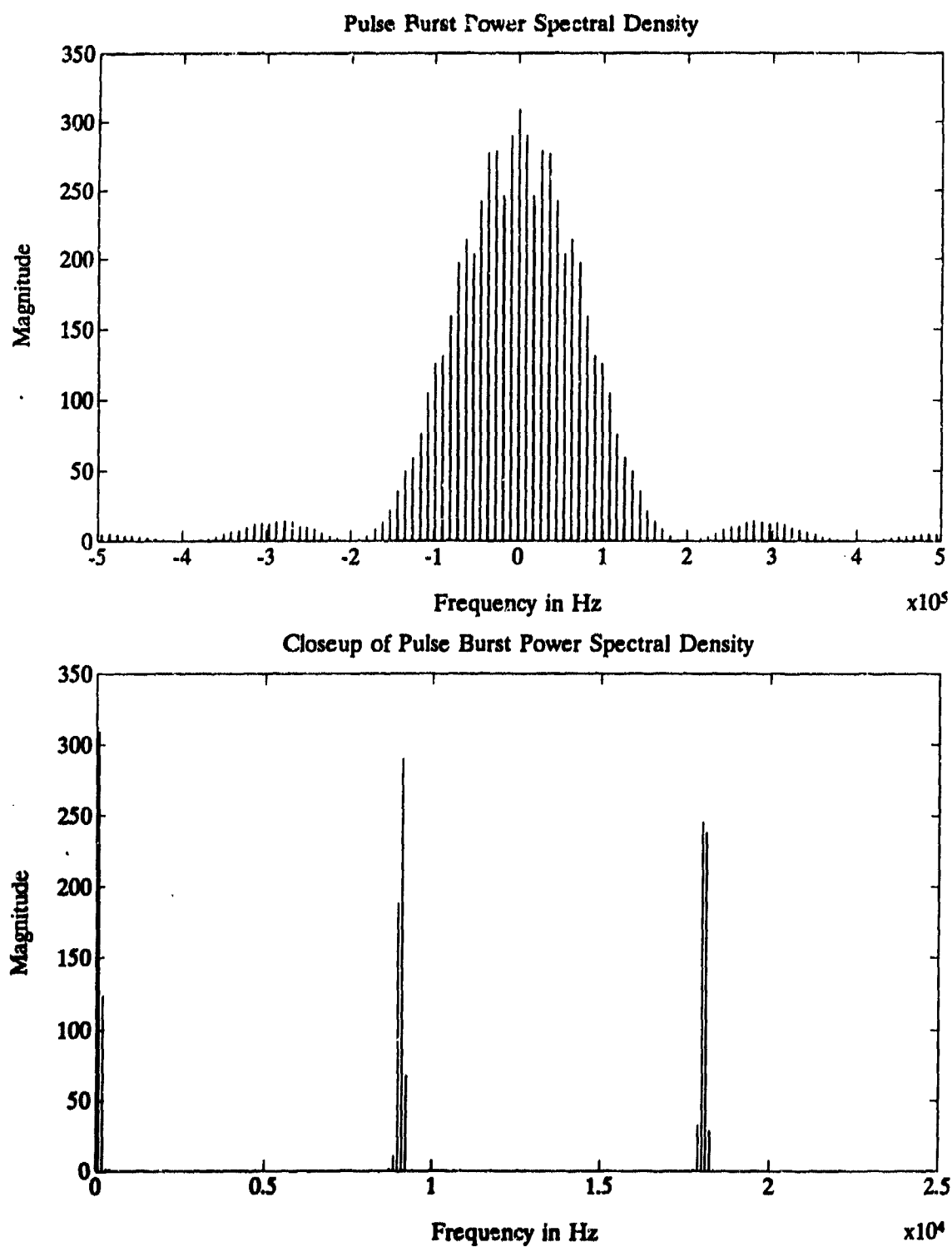


Figure 6. Pulse Burst Spectrum (Sample Rate = 1 MHz)

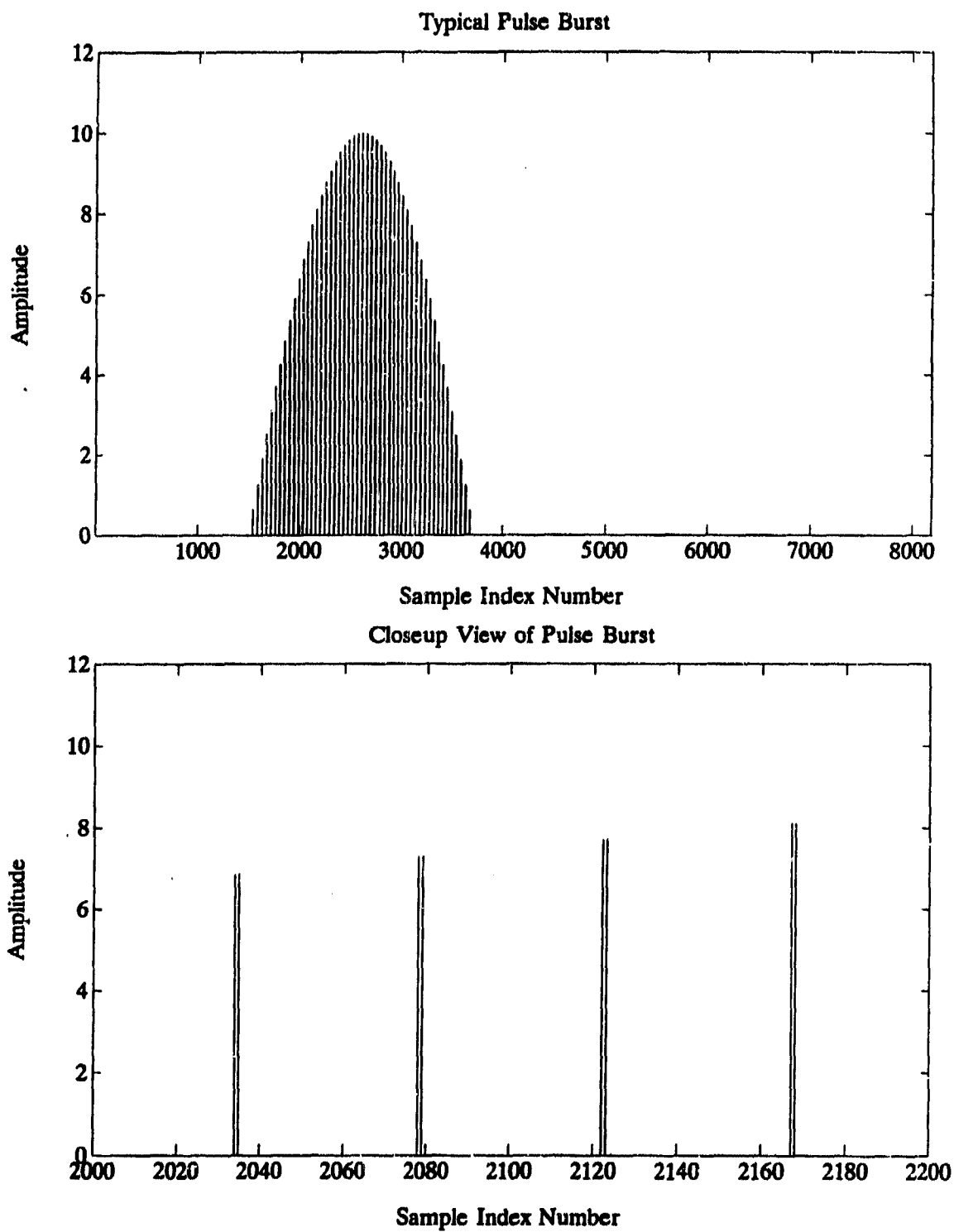


Figure 7. Pulse Burst (Sample Rate = 400 kHz)

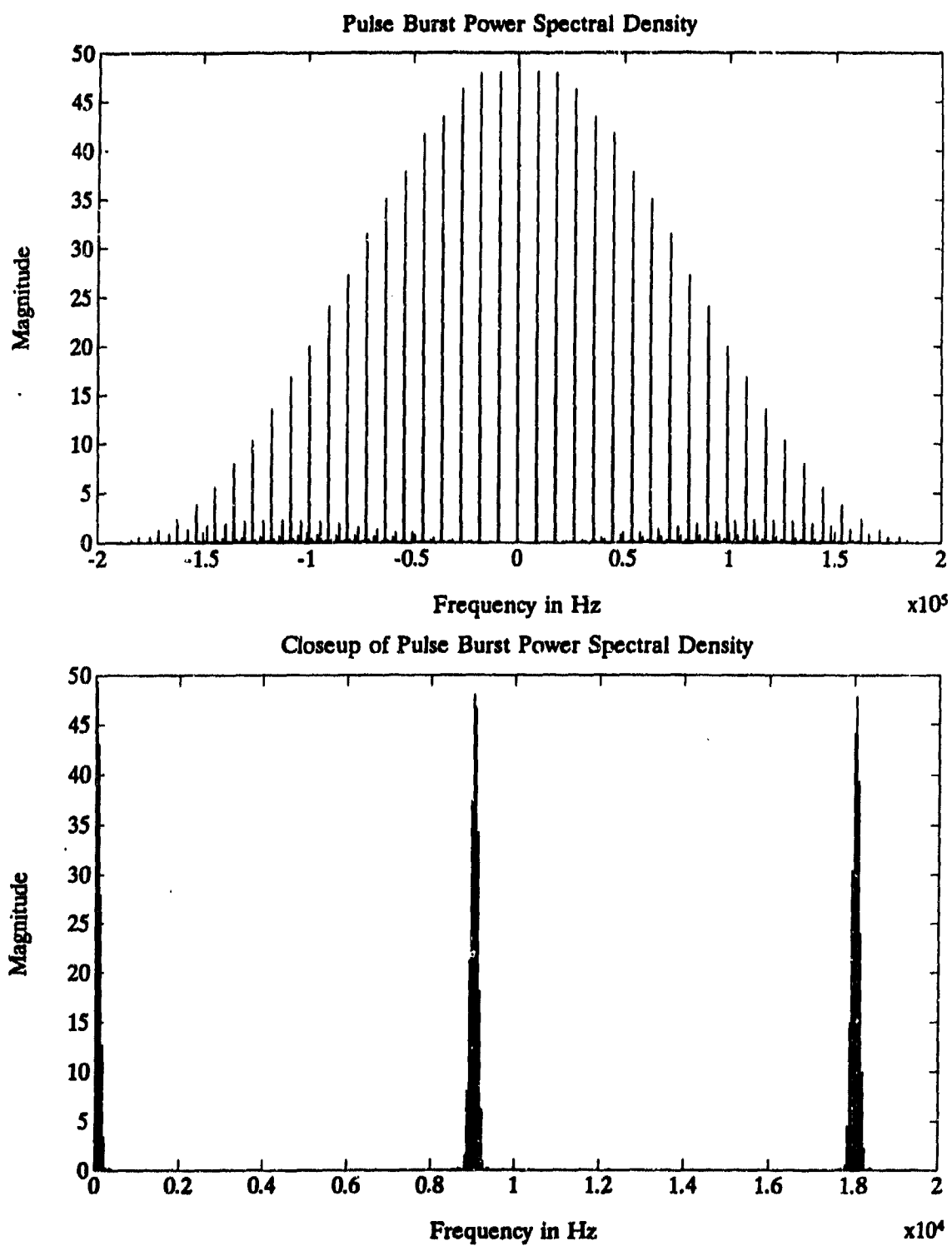


Figure 8. Pulse Burst Spectrum (Sample Rate = 400 kHz)

IV. TIME DOMAIN BASED SIGNAL ANALYSIS

One of our goals was to produce a method for detecting and locating medium to high PRF signals. The method developed uses a decision algorithm to process raw, time domain radar data. Estimates of the values of PRF and scan rate are smoothed using Kalman filtering techniques.

The Fast Fourier Transform is used during the initial phase of acquiring a emitter in order to produce estimates of the received SNR as well as providing a check on the time domain measurement of PRF. The remainder of the processing is carried out in the time domain. Optimal estimation theory (optimal filtering of signals embedded in noise) is based on time domain methods, as are many aspects of modern control theory (i.e., optimal control and adaptive control [Ref 5.]).

All of the source code files for these algorithms are given in Appendix B. No discussion of the coding of the algorithms is presented in this text other than comments included in the source code files themselves.

A. PRF/PRI MEASUREMENT

The algorithm used to process the radar data depends on a floating threshold. Ideally, we would discuss this first. However, it will be much easier to assume that the threshold value is known a priori and develop the measurement techniques. Once it has been shown how the signal parameters are obtained, the actual method for setting the threshold will be discussed. The PRF/PRI measurement forms the central process in the

overall method. Every incoming bit in the data stream must be processed to find out if a pulse has occurred. If it has, then a flag is set and the time between pulses is retained. This forms the basis of an estimate of the PRF of the radar emitter. Further logic allows us to measure the scan rate and to produce an estimate of the burst envelope to be used for the TOA problem. A first order Kalman estimator is employed to smooth out individual estimates of the signal PRF.

As the pulse stream enters the processor, each sample is checked to determine if it exceeds a threshold value. When two samples in succession exceed threshold, then the algorithm declares that a pulse has occurred. The time between detected pulses serves as an observation of the PRI. Since the PRI is assumed to be constant for a specific emitter, the following equations serve as the system model for the PRI [Ref. 6]:

$$\begin{aligned}\hat{x}_{k+1} &= \hat{x}_k; \\ z_{k+1} &= z_k + v_k;\end{aligned}\tag{10}$$

where

\hat{x}_k = PRI estimates,
 z_k = PRI observations, and
 v_k = observation noise with variance r_0 .

The discrete optimal filter for this system is

$$\hat{x}_{k+1} = \hat{x}_k + \frac{1/r_0}{k+1/r_0} (z_k - \hat{x}_k).\tag{11}$$

If the noise v_k is Gaussian noise with zero mean and $r_0=1$ the optimal filter is

$$\hat{x}_{k+1} = \hat{x}_k + \frac{1}{k+1} (z_k - \hat{x}_k). \quad (12)$$

In the simulation, we have assumed that the sampled data is corrupted by Gaussian white noise of zero mean and unity variance. The noise out of the envelope detector would, of course, be Rayleigh distributed. Gaussian noise is used as a simplifying assumption. The addition of Gaussian noise to the signal data can also be interpreted as modelling noise sources other than the background noise (i.e., thermal noise in the receiver past the envelope detector).

It was noted during development that the noise rejection of the method was not adequate. Once the algorithm detects the start of a pulse burst, it changes its operation to begin estimating the parameters of interest. After the first detected pulse, the next several will be susceptible to being 'dropped.' Dropped pulses are those pulses which are not detected and are inside the region of the pulse burst that the method integrates to determine energy centroid for use in the TOA problem.

There are several mechanisms which can lead to dropped pulses. Due to the presence of noise one of the samples inside a pulse might be below threshold. It is possible to obtain only one sample during a pulse if the sample frequency is too low or if the radar pulse is short. Radars are susceptible to skipping pulses. Therefore, there may not have been a pulse from the emitter. Nothing can be done about pulses that the emitter never sends. If these are infrequent events, the Kalman estimation will still be able to produce good results.

If a pulse is dropped, the value of the residual in the above equation becomes large enough that the PRI observations will be in error. Remember that a dropped pulse yields an observation with a relative error of 100 percent. Since there are relatively few pulses per burst, (typically from 90 to 150 pulses exceeding threshold), a single dropped pulse per burst becomes a significant error.

To eliminate dropped pulses due to noise problems, there are two possible solutions. First, the sample rate can be increased to produce more than two samples per pulse and more complicated detection routines can be employed. Another, simpler, solution is to set a truncation value. The first n sample pairs which exceed threshold are not used. The filter counter starts only after n detects. Since dropped pulses occur most frequently at the beginning and end of a pulse burst, the last n pairs in the pulse burst are also discarded. This method significantly reduces the number of dropped pulses. At present, the truncation value must be set by the operator. Ideally, this would be set by a CFAR (constant false-alarm rate) method [Ref. 4]. The truncation value doubles as a flag that the burst is over. When n expected detects have been missed, processing of the data is suspended until just before the next, expected burst.

It might seem that we are relying on a substantial amount of a priori knowledge about the signal to make this method work. Often there are specific known signals which are of interest, and accurate tailoring of the detection scheme to the emitter signal is an effective way to improve its performance. In addition, all that is really required is a good estimate of PRI. This comes from a single batch FFT process on the initial gain of a

emitter. In the future, the batch FFT could be used repetitively as a second check on the Kalman estimators.

We are performing analog-to-digital conversion of the envelope detected signal at a rate of two to three times the reciprocal of the pulse width. However, other than simple threshold logic, we are not processing data at this rate. The algorithm actually samples the digital pulse train at only two to four times the PRF once it has been synchronized with the pulses. This method of PRI measurement appears promising with proper glitch rejection and dropped pulse logic.

B. SCAN RATE MEASUREMENT

Scan rate measurement stems from the PRI detection method. The first detect marks the beginning of a burst; the last marks the end. Processing these numbers allows calculation of the burst duration (and, therefore, the azimuthal bandwidth of the emitter). The distance from the start of one burst to the start of the next can be processed to give scan period. We are only looking for acquisition radars which have circular scans in azimuth. The scan rate is the number of 360° scans per unit time. Typical values for this would be 1 scan per second.

As stated, the data must be sampled at a little more than twice once over the pulse width. This ensures at least one sample inside each pulse in the burst. A higher sampling frequency allows for more samples per pulse and would allow for pulse shape analysis. However, this is not the goal of this method, and the desire to minimize the amount of

data handling is strong. Minimizing the data required on an individual emitter, increases the number of possible emitters which the system could process.

The time is also retained when the pulse starts. The next burst determines the initial estimate of scan rate. Scan rate is a constant which can be modelled as a digital system using the same system model as PRF:

$$\begin{aligned}\hat{x}_{k+1} &= \hat{x}_k; \\ z_{k+1} &= z_k + v_k;\end{aligned}\tag{13}$$

where

\hat{x}_k = scan rate estimates
 z_k = scan rate observations, and
 v_k = observation noise with variance r_0 .

The development proceeds as before, and the final form of the filter is

$$\hat{x}_{k+1} = \hat{x}_k + \frac{1}{k+1} (z_k - \hat{x}_k).\tag{14}$$

Scan rate is assumed to be constant here. If the scan rate changes, it would be easy to impose limiting values for the change and prompt the operator when these are exceeded. The change could signal switch from scan to track mode as discussed earlier, a lost emitter contact, or the gain of a new emitter.

Once the first estimate of scan rate has been made, the algorithm no longer needs to test every bit coming in. The program can use the initial estimate to narrow the range of data it processes in order to make less computation necessary to sort information from

the data. At this point the algorithm turns itself on at 90% of the expected scan time and begins processing again.

C. TIME OF ARRIVAL

During the PRF and scan rate measurement, it is possible to use an additional Kalman filter to produce a smoothed approximation to the burst envelope. The two sample detection scheme to identify and count pulses provides us with a means to begin to predict the shape of the pulse envelope. On initial detection of a pulse burst, the filters are initialized with discrete counters in order to allow processing on a digital computer. The observation vector consists of the two discrete samples which were used to detect the pulse. The magnitudes of the samples are summed and treated as a single estimate. The filter, since we are using a first order approximation, is developed in the same manner as before. This procedure of summing the two samples used to detect a pulse to form an observation z_k is an *ad hoc* development. The final form is

$$\hat{x}_{k+1} = \hat{x}_k + \frac{1/r_0}{k+1/r_0}(z_k + z_{k-1} - \hat{x}_k). \quad (15)$$

where

- z_k = individual samples in the pulse stream,
- r_0 = the noise variance, and
- \hat{x}_k = estimates of envelope shape.

The counter k is only updated when observations occur, i.e., two consecutive samples exceed threshold. These estimates of envelope must be retained for the duration of a pulse burst to allow the TOA algorithm to calculate the energy centroid. The set of

estimates provides an approximation to the envelope which is integrated numerically in order to find the centroid of the shape. This centroid is what gives us the estimate of signal time of arrival.

D. SNR AND THRESHOLD DETERMINATION

To provide a measure of the signal-to-noise ratio is very important. If intelligence regarding the source strength of the emitting platform is known, then the signal strength can be used as a first approximation to the emitter range. Prior to the initial detect sequence, FFT methods are used to process the noise. The discrete Fourier transform is defined as [Ref. 6]

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)nk} . \quad (16)$$

The zero order coefficient is

$$X(0) = \sum_{n=0}^{N-1} x(n) . \quad (17)$$

Dividing by the number of samples gives us the mean:

$$\text{mean} = \frac{X(0)}{N} = \frac{\sum_{n=0}^{N-1} x(n)}{N} . \quad (18)$$

This method is convenient since the FFT of the data is also used to obtain initial estimates of the PRF. The mean along with the noise variance, can be used to set a

threshold value. A good threshold can be selected using detection theory concepts (i.e., CFAR detection):

$$\text{threshold} = \text{mean} + a r_0; \quad (19)$$

where

a = a floating value set by CFAR methods, and
 r_0 = the noise variance.

The FFT is next used to calculate the power spectral density of the data:

$$S(k) = \frac{X(k) \times X(k)^*}{N} . \quad (20)$$

The total power in the data is

$$P_{\text{data}} = \sum_{k=0}^{N-1} S(k) . \quad (21)$$

To estimate the SNR, the noise and signal powers are found separately. FFT's are taken in batches of 1024 for the noise only segments and 8192 for the signal segments. The measured and reported SNR is

$$\text{SNR} = 10 \log \left(\frac{P_{\text{signal}}}{8 \times P_{\text{noise}}} \right) . \quad (22)$$

These measurements and methods form the preprocessing in the area of extracting real time position information on emitting emitters and using Kalman optimal filtering techniques in order to provide emitter locations.

V. PERFORMANCE RUNS AND RESULTS

This chapter shows graphical results of the methods described here. All the simulations were performed on a personal computer or a UNIX workstation. The real implementation of this technique would be such that, data enters as a stream of digitally sampled values from an analog to digital converter. The memory requirements are small, and only enough samples need be retained for the 8192 samples need to be retained in the batch FFT. To simulate this situation on a personal computer requires that a variable be loaded with the values for received data or that the computer manufacture the data "on the fly." Since all of the signal data is stored, the first situation can lead to memory problems during simulation which would not arise in a real implementation of this work. Even at moderate sample rates, the total amount of data used by the method quickly becomes large. The second method does not allow for displaying the data to reconstruct events which caused the algorithm to produce poor results.

Due to the limitations described above, the emitter parameters were manipulated to improve the efficiency of the simulations. To permit more scans in a shorter period of time (fewer total data samples), the scan rates were set much higher than would be reasonable. The beamwidth of the emitter was similarly increased to provide the proper number of pulses per burst. Receiver velocity has been exaggerated to allow receiver movement to be seen during the short time span of the simulations. Therefore, values for parameters other than PRI will not be realistic.

A. EMITTER WITH ZERO NOISE

To test the algorithm, a zero noise emitter was created. If the algorithm does not accurately predict the parameters of a noise-free pure signal, there is a fundamental error which must be corrected. Figure 9 shows the data used. There are five pulse bursts present. This data is the pulse train which will be seen by the first receiver. Data for the second receiver has the individual bursts shifted in time. The amount of shift is based on the emitter and receiver geometry. Factors such as Doppler shift due to the moving receiver are ignored. Data is manufactured so the precise time delays necessary for good angles of arrival exist in the data. If the algorithm finds the centroids properly, then the bearing lines to the emitter will be exact.

The data in Figure 9 serves as the basis for all the tests presented here. For the case where noise is present, the data simply had zero mean, Gaussian noise added. The important parameters are:

- Sample Frequency - 400 kHz
- PRF - 9000 Hz
- Scan Rate - 35 Scans/second
- Beamwidth - 200°
- Receiver velocity - 300,000 km/hr

The test on the zero noise emitter was successful and plots of the results can be seen in Figures 10-12. Since there is no noise, the algorithm uses a threshold and a truncation value of one. There will not be any dropped pulses in this emitter. Figure 10 shows the PRI history at both receivers. There is a bias in the Kalman filter of approximately 3 Hz. The source of the bias is, as yet, unknown. The final value of PRF

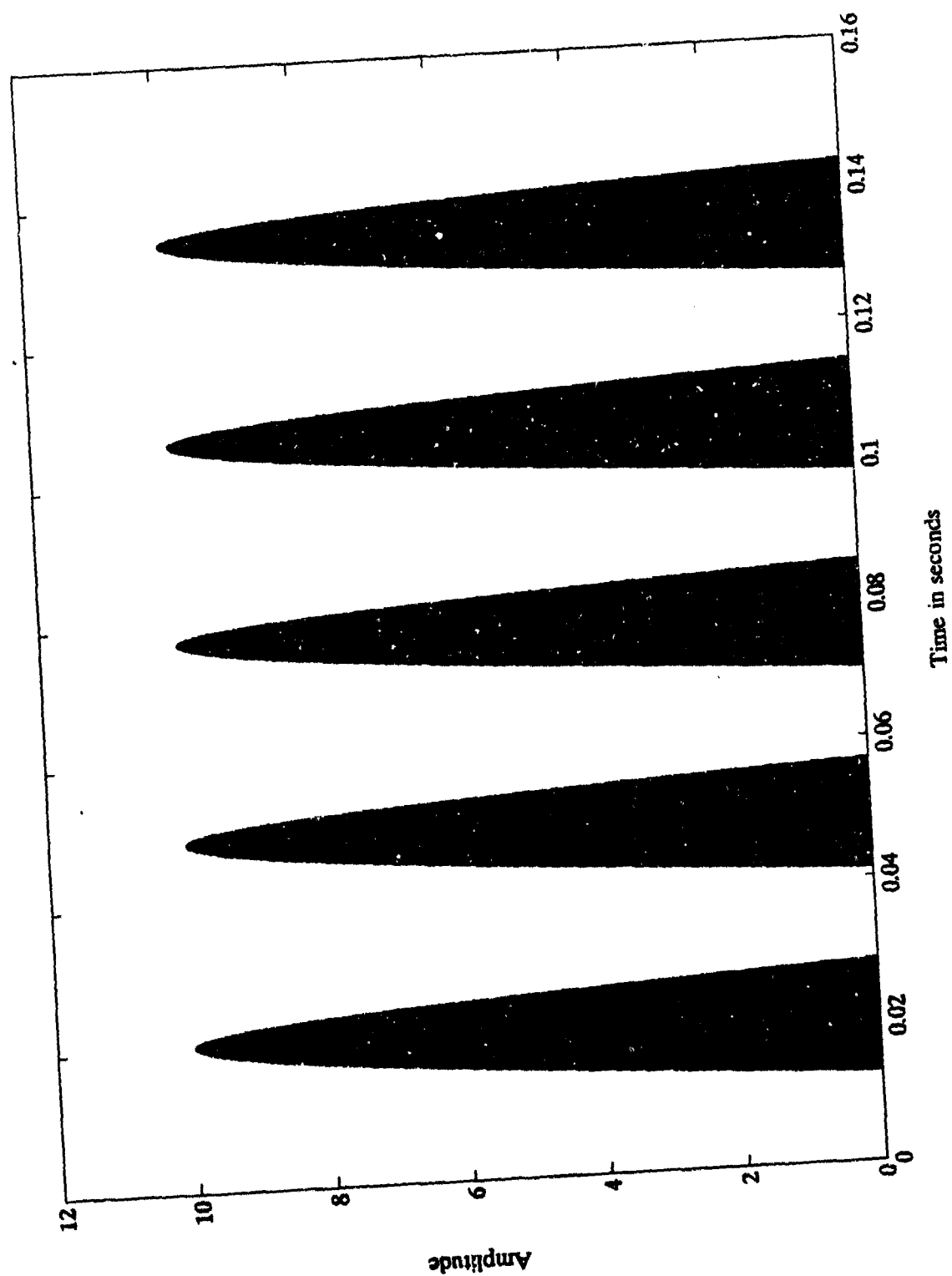


Figure 9. Emitter with Zero Noise

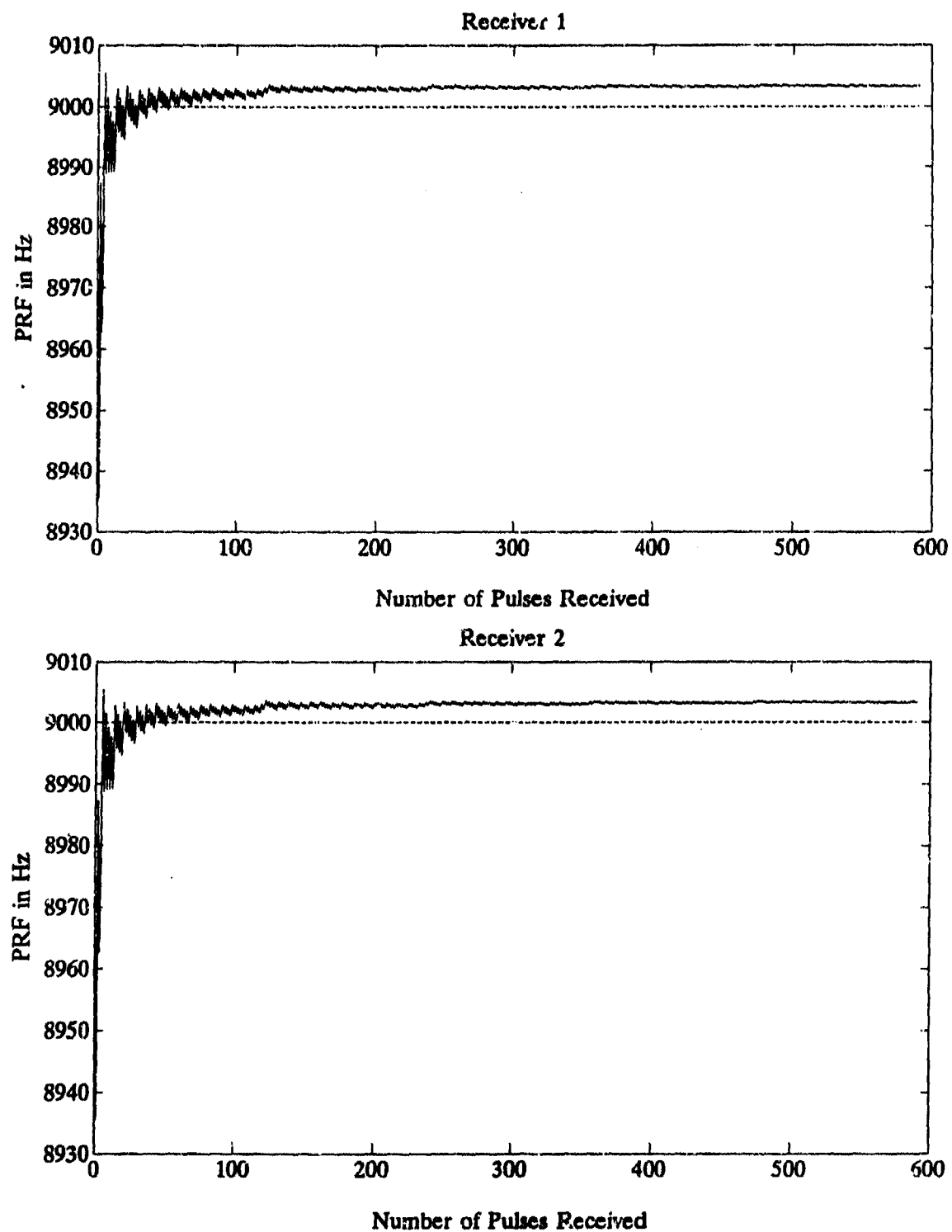


Figure 10. PRF Histories for the Zero Noise Emitter

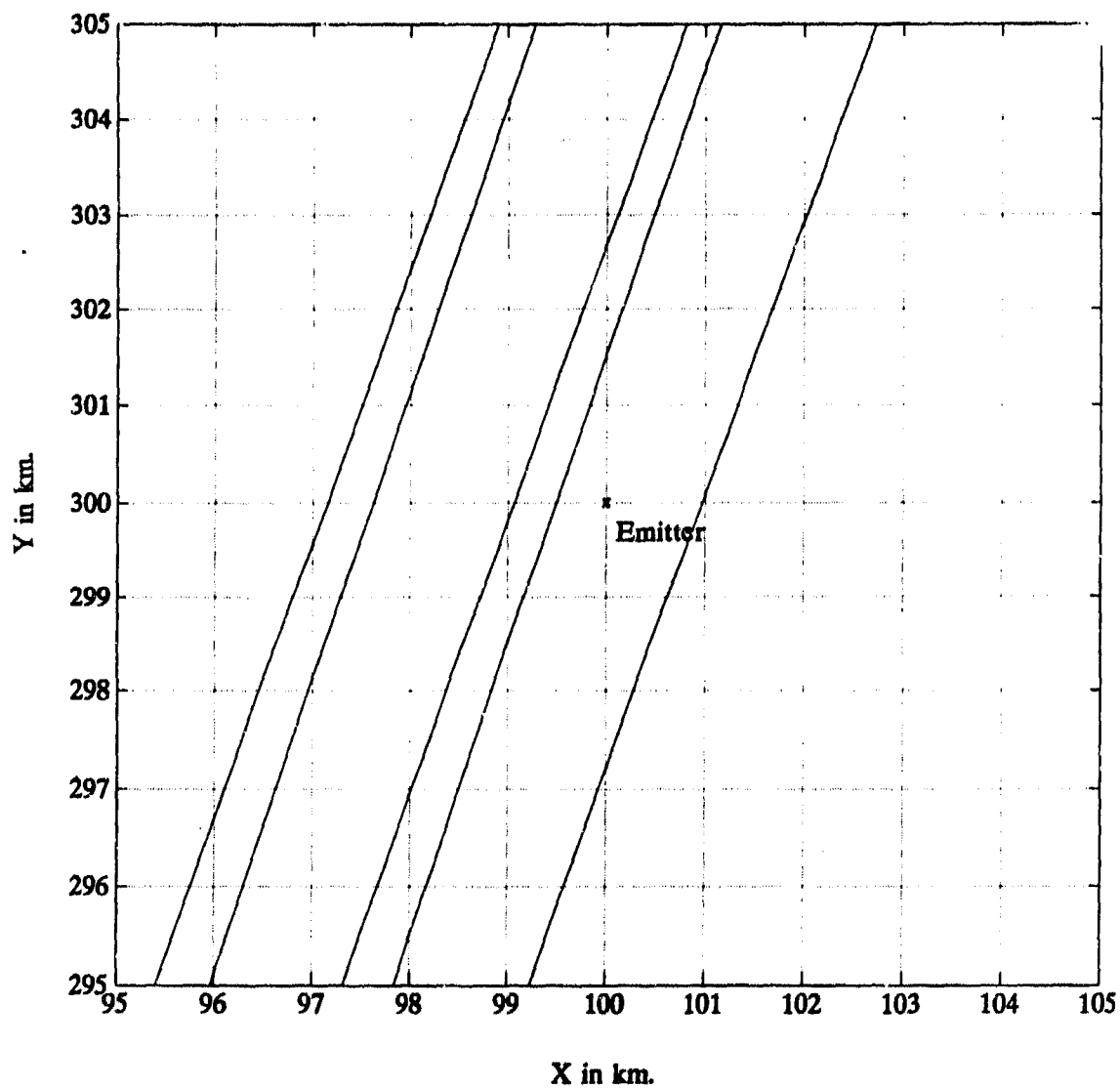


Figure 11. AOA Estimates for Zero Noise Emitter

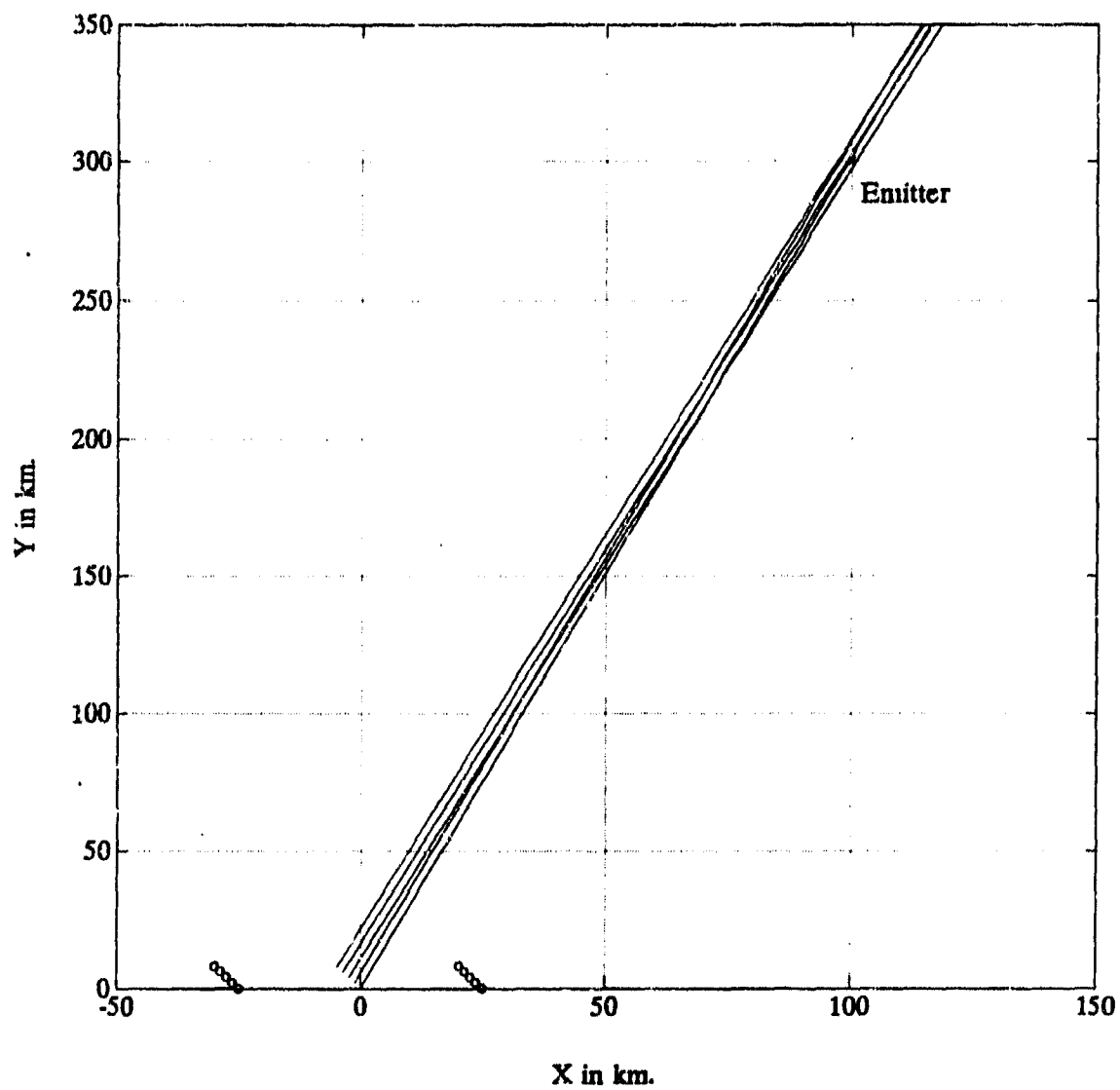


Figure 12. Closeup of AOA Estimates for Zero Noise Target

returned by the algorithm was 9003 Hz. This is better than the estimate from the batch FFT process. This estimate was 8984 Hz. This value served as the initial estimate for the Kalman filter. The reason for the FFT's error can be understood by looking at the resolution of the FFT:

$$\Delta f = \frac{\text{Sample Frequency}}{N}; \quad (23)$$

where

N = the number of samples in the batch FFT.

The Δf for the FFT of 8192 sample used is about 48 Hz. The Kalman filtering method provides superior resolution even with the current bias problem. The PRF estimate comes from looking for the maximum value in the FFT output. Some of the error from the FFT can be removed by averaging the results of FFT bins near the maximum.

Scan rate histories are not shown. With only 4 to 6 pulse bursts in a typical simulation run, there is not much data to be seen. The probability of missing a burst is zero (for the simulation data) and the bursts are far enough apart in time, that the dropped pulse problem does not seriously affect the scan rate estimates.

Figures 11 and 12 show the results of the angle of arrival (AOA) portion of the algorithm. Receiver positions over time are shown as circles in the lower portion of the graph. The lines show the estimated angles of arrival to the emitter for each set of receiver positions. For this simulation it is assumed that the receivers are a fixed distance apart and are traveling at a constant velocity. Figure 12 is a closeup view of the bearing lines in the vicinity of the emitter. Obviously, the estimated angles are sufficient to

predict the emitter bearing. Due to the narrow crossing angles of the bearing lines over time, position estimates of the emitter are not possible for these simulations. Even at the huge simulation velocity, the emitter bearings do not change much over time. Longer simulations using a different method generating emitter data will be required to produce sufficient numbers of emitter angles for tracking.

B. EMITTER WITH ADDED NOISE

Figure 13 shows the same emitter data as in Figure 9, but with added noise. Figure 14 shows the PRI history using the program's a calculated threshold of 3.118 (calculated by Equation (19) with $a=3$) and a truncation value of 8. The effect of dropped pulses is quite clear in the results of this run. Pulses detected at the trailing edge of a burst reset the burst processing algorithm and an entire scan period becomes an estimate of PRI, this accounts for the large spikes in the graph for receiver 2.

Figures 15 and 16 shows two more trials on the same file with adjusted threshold and truncation settings. Figure 15 again uses the threshold from Equation (19), but a truncation value of 12. Results for receiver 1 are improved, but some event disrupting the method has not been truncated for receiver 2. For the final run, the threshold was boosted manually to 3.5 (the calculated setting was 3.118) and truncation value was left at 12. Good performance can be seen at both receivers. At present, adjustment of these values is not automatically performed by the method. A study of adjusting these values based on a CFAR method would be useful.

Figure 17 gives the estimated bearing lines. Due to insufficient noise rejection by the algorithm, the results are poor. The estimates are only good enough to locate the quadrant of the emitter and not its actual position. More work is needed on the filtering and integration process which produce the estimated burst envelope and its centroid.

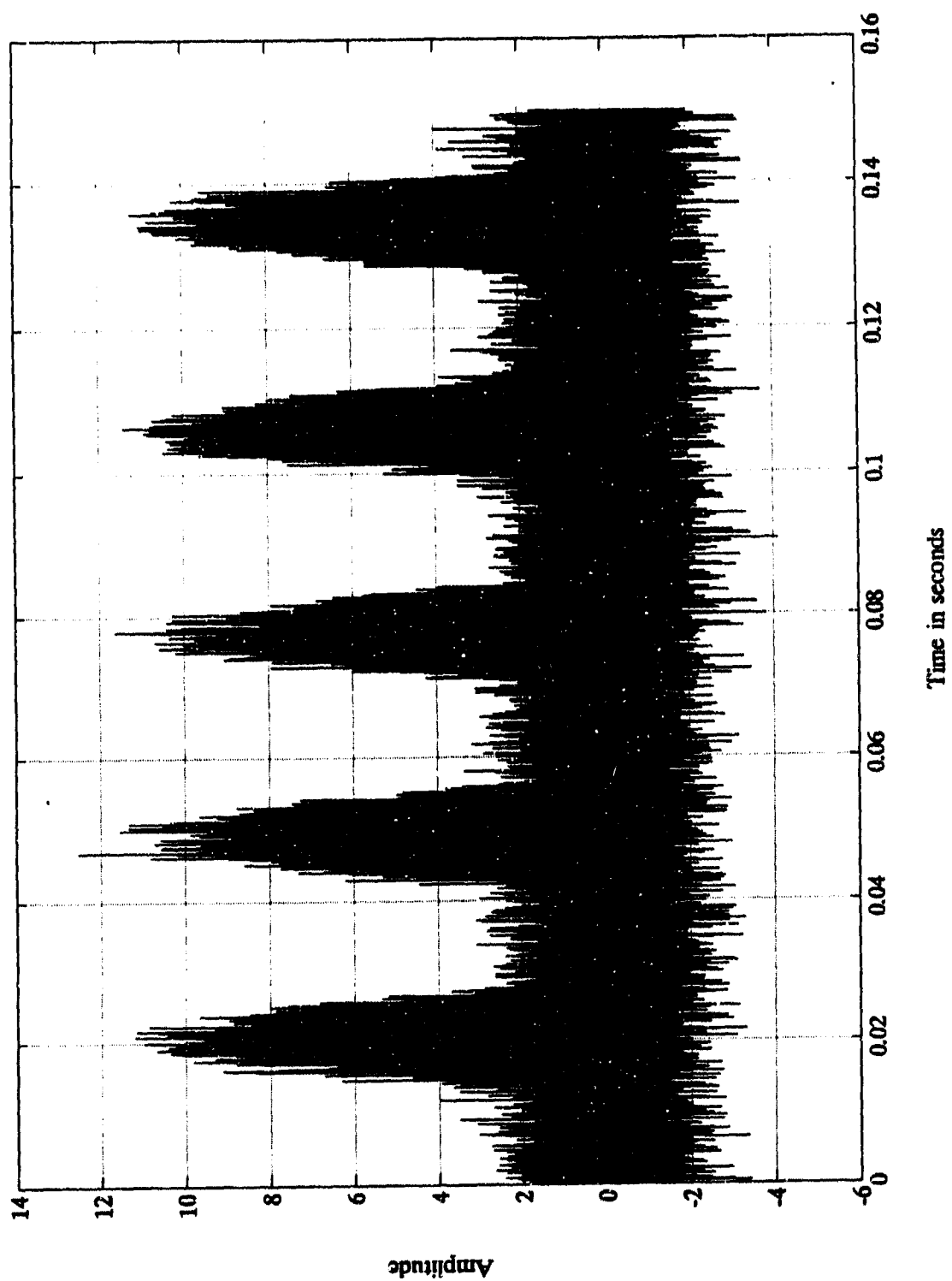


Figure 13. Emitter With Gaussian Noise

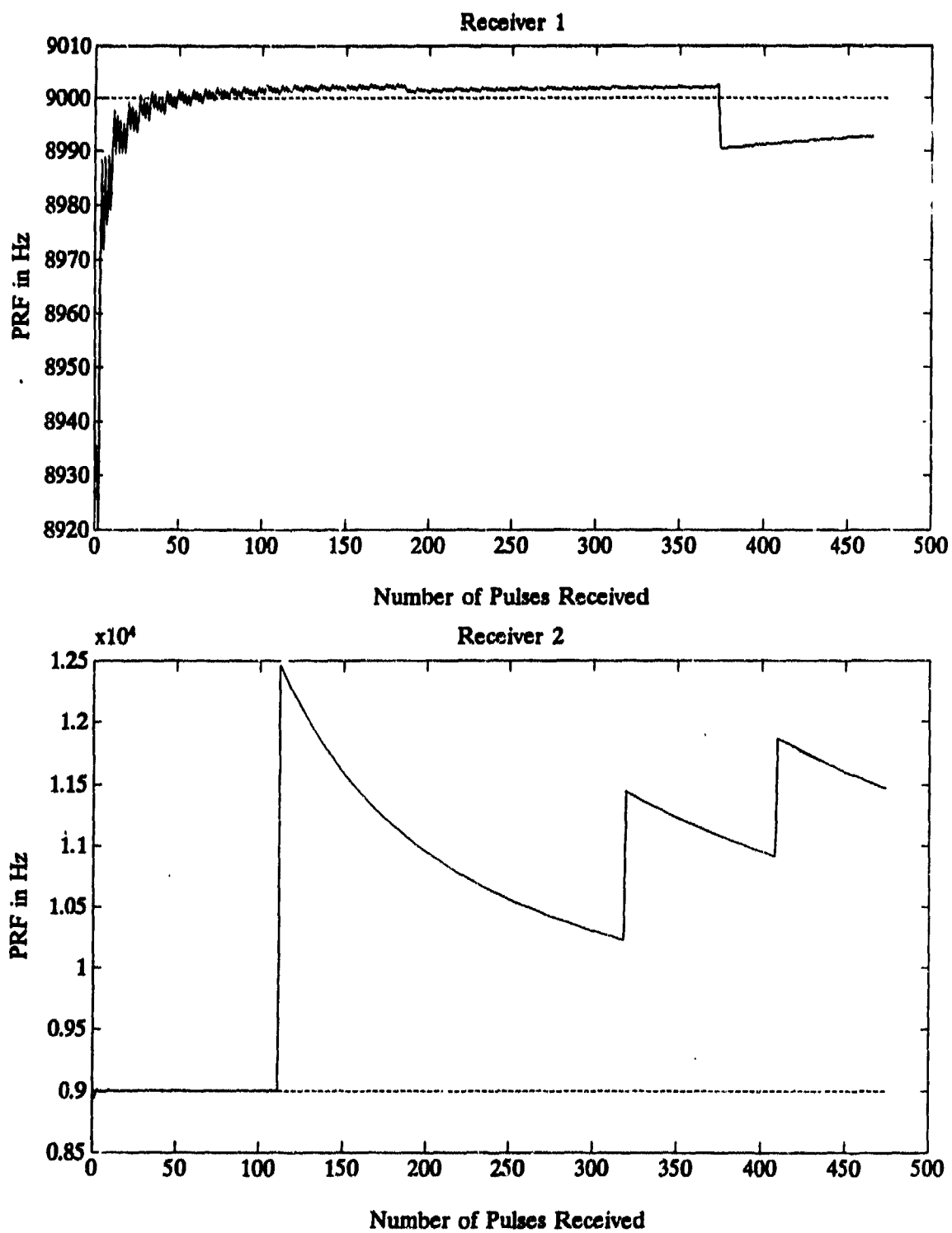


Figure 14. PRF History for Trial #1

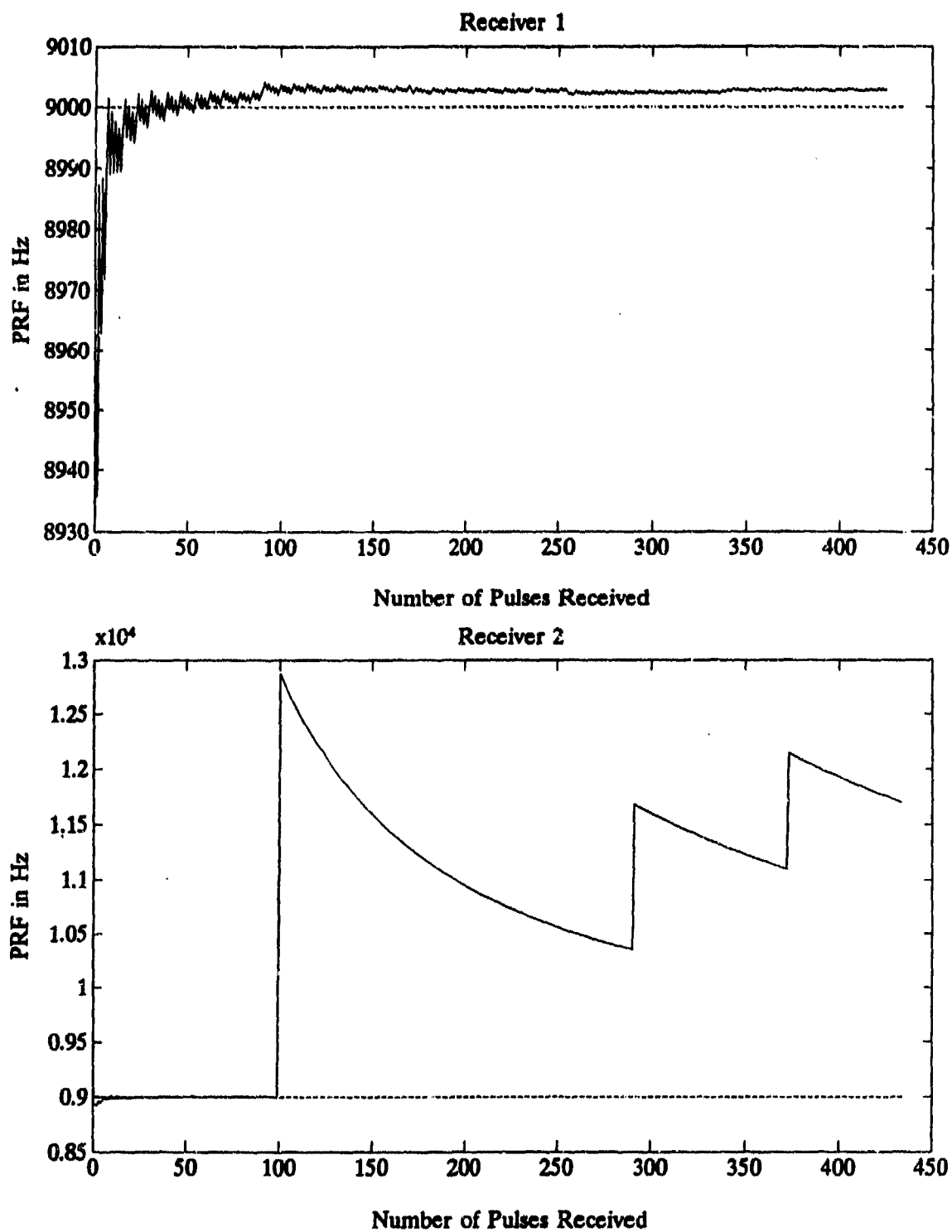


Figure 15. PRF History for Trial #2

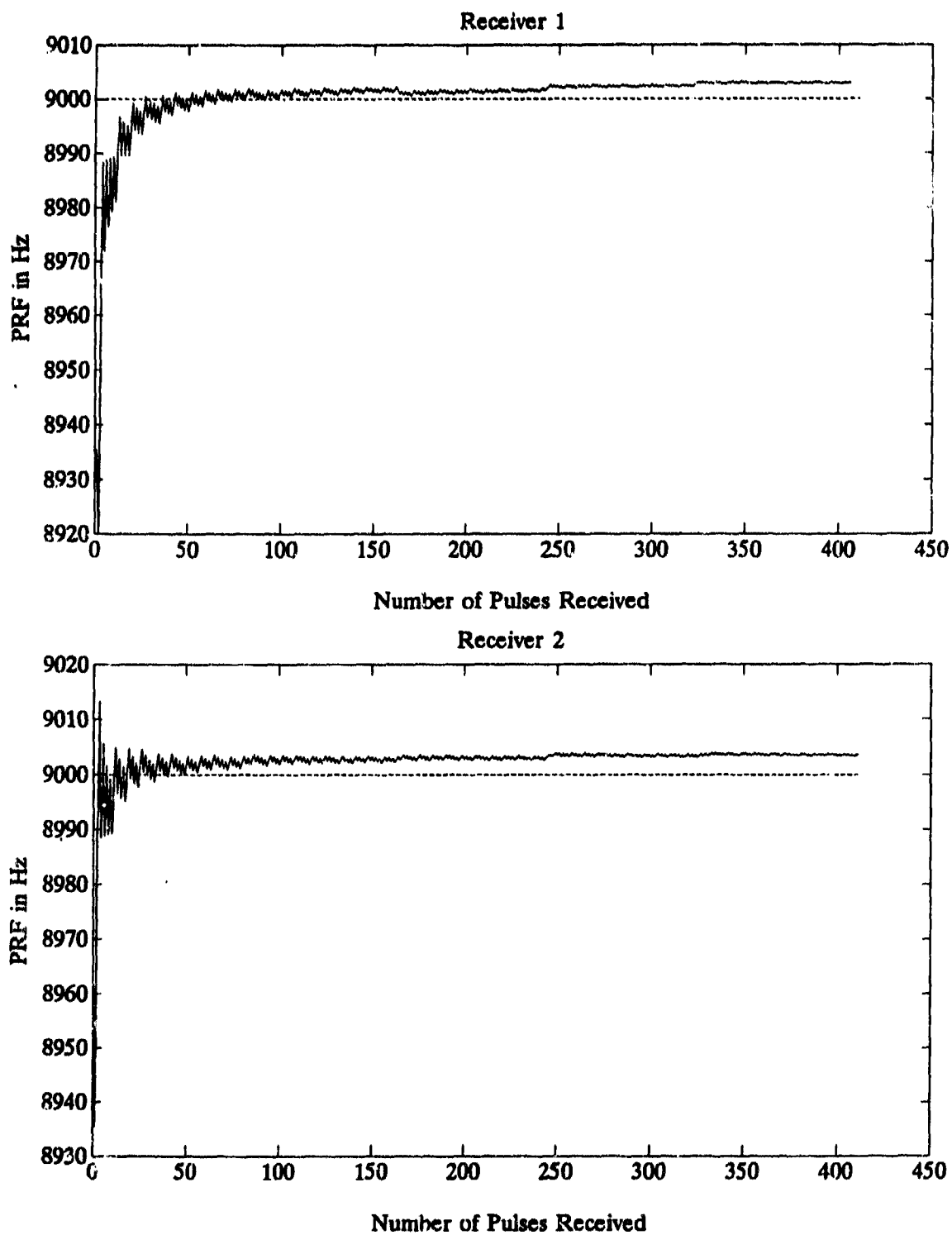


Figure 16. PRF History for Trial #3

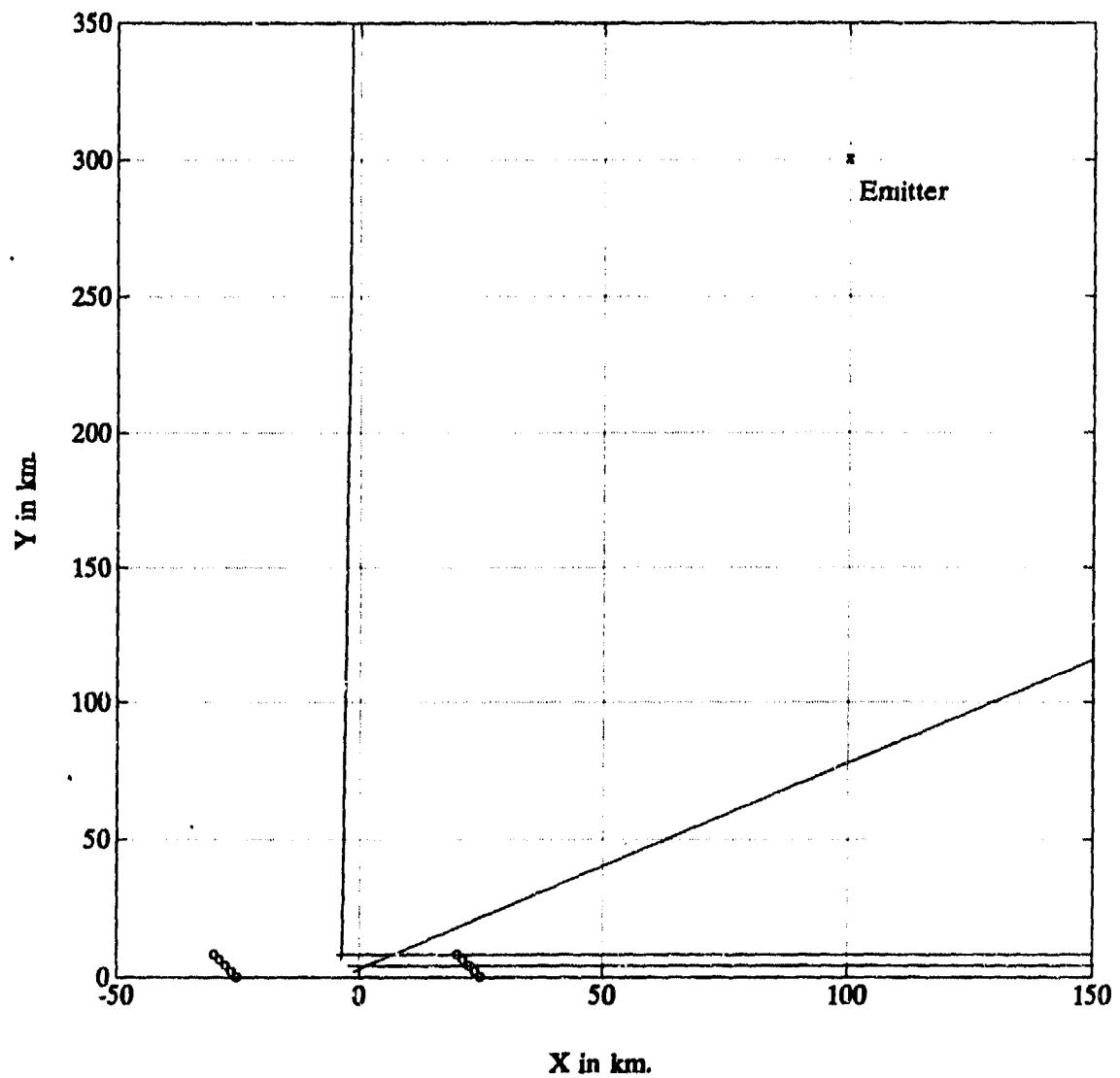


Figure 17. AOA Estimates for Noisy Emitter

VI. SUMMARY

A. CONCLUSIONS

Further refinement of the method presented here and more work on classified data is in progress. The PRI estimates produced by the method are very accurate. These estimates are highly sensitive to dropped pulses. With the proper choice of threshold and truncation values, false or dropped detects were eliminated. Scan rate estimates were also good. The one significant drawback of the work done here is that it does not allow for easy application of itself to a radar or geometry that is different. The technique is precisely tailored to a particular type of emitter. However, having special algorithms can greatly improve the ability to detect specific emitters.

B. RECOMMENDATIONS

The author's suggestions for other research and points of interest occur sporadically throughout the text. Some final comments will be made here. The SNR of simulated data tested was high. This allowed testing and simulation to occur more easily. No research into the minimum SNR that can be analyzed by these techniques has been performed. Since simulations were performed on rather small data sets, the batch FFT processing occurs only on the first detected pulse. It is the author's contention that the FFT of every pulse burst is not necessary. More research should be performed to determine how often to reset the algorithm and update the noise measurements.

The algorithm presented currently has very poor resolution for the angle of arrival of the signal at low SNR. The method was developed with speed and minimum data storage as the primary goals. Improving the resolution could come by several means. Increasing the data rate will allow many more samples inside the burst and should increase the resolution at the cost of increased processing requirements.

APPENDIX A. ELINT RANGE ADVANTAGE

Previously in this work, the fundamental range advantage of ELINT over radar was mentioned briefly, but never fully developed. This appendix will provide a more detailed approach to the range advantage of ELINT receivers. The equations and general flow of this derivation are from [Ref. 1].

Consider the scenario of an emitting radar and an ELINT receiver attempting to intercept the radar signal. The signal power at a radar receiver is given by the basic radar equation [Ref. 1]:

$$S_R = \frac{P_T G_T G_R \lambda^2 \sigma}{(4\pi)^2 R_R^4 L_T L_R} ; \quad (24)$$

where

- S_R = signal power at radar receiver (watts),
- P_T = transmitted power (watts),
- G_T = gain of radar transmit antenna,
- G_R = gain of radar receive antenna,
- λ = wavelength,
- σ = emitter radar cross section (m),
- R_R = radar losses from transmitter to antenna,
- L_T = radar losses from the transmitter to the antenna, and
- L_R = radar losses from antenna to receiver.

The ELINT receiver, however, does not rely on the reflected energy from an emitter and so we use the one-way or "beacon" radar equation [Ref. 1]:

$$S_E = \frac{P_T G_{TE} G_E \lambda^2}{(4\pi)^2 R_E^2 L_T L_E} ; \quad (25)$$

where

S_E = signal level at the ELINT receiver (watts),
 G_{TE} = gain of radar transmit antenna in the direction of the ELINT receiver,
 G_E = gain of the ELINT antenna,
 R_E = range from radar to ELINT receiver,
 L_E = loss from ELINT antenna to receiver, and
all other symbols are as they appear in Equation (24) above.

Suppose that the signal level required at the ELINT receiver is a factor of δ times the signal level required at the radar to detect the emitter:

$$S_E = \delta S_R \quad (26)$$

Suppose, also, that the range, R_E , at which the ELINT received power level is δS_R is a multiple of α times that range at which the radar received power level is S_R :

$$R_E = \alpha R_R \quad (27)$$

Substituting Equations (26) and (27) into Equation (25) above, gives [Ref. 1]:

$$S_R = \frac{P_T G_{TE} G_E \lambda^2}{\delta (4\pi)^2 R_R^2 \alpha^2 L_T L_E} \quad (28)$$

By substituting Equation (28) into Equation (24) and solving for α we obtain [Ref. 1]

$$\alpha = \frac{R_E}{R_R} = R_R \left[\frac{4\pi}{\delta} \cdot \frac{1}{\sigma} \cdot \frac{G_{TE} G_E}{G_T G_R} \cdot \frac{L_R}{L_E} \right]^{\frac{1}{2}} \quad (29)$$

Suppose, for convenience, that the losses in the radar and ELINT receivers are equal.

Suppose also that the radar's transmit and receive antenna gains are equal and that the ELINT receiver is in the side lobe of the radar such that $G_{TE} = 1$. Finally, suppose that the ELINT receiver antenna is omnidirectional ($G_E=1$). Then Equation (29) becomes

$$\alpha = \frac{R_R}{G_R} \left[\frac{4\pi}{\delta \sigma} \right]^{\frac{1}{2}}. \quad (30)$$

The factor $(1/\delta)^{1/2}$ is the advantage enjoyed by the radar due to its more sensitive receiver. (The radar receiver can be matched to its signal and can use multiple pulse integration.) The ELINT receiver generally has a wider noised bandwidth than the radar receiver and operates on a single pulse basis. Therefore the minimum that δ can be is one and it is typically as large as 100 or 1000. The radar antenna gains are typically in the 30 to 40 dB range. Figure 18 shows Equation (30) plotted for several values of δ and G_R . Figure 18 also shows the main beam intercept case, in which $G_{TE} = G_R$ for this case, the ELINT range advantage is given by

$$\alpha = R_R \left[\frac{4\pi}{\delta \sigma G_R} \right]^{\frac{1}{2}}. \quad (31)$$

Let us assume that our ELINT receiver is in the side lobes of a radar having a 40 dB gain antenna, which can detect a 1 m² emitter at 100 km. If the ELINT receiver is 20dB less sensitive than the radar receiver ($\delta = 100$) then the ELINT range advantage from Equation (30) is

$$\alpha = \frac{10^5 m}{10^4} \left[\frac{4\pi}{100 \times (1 \text{ m}^2)} \right]^{\frac{1}{2}} \approx 3.5. \quad (32)$$

This is a substantial advantage over the radar. If we move the receiver into the main beam then, for the same radar and ELINT receiver, Equation (31) gives the new ELINT range advantage:

$$\alpha = 10^5 m \left[\frac{4\pi}{100 \times (1 \text{ m}^2) \times 10^4} \right]^{\frac{1}{2}} \approx 350. \quad (33)$$

This means that we should be able to detect the radar at a range of 35,000 kilometers. This is, of course, only a theoretical detection range and does not account for such effects as line-of-sight restrictions.

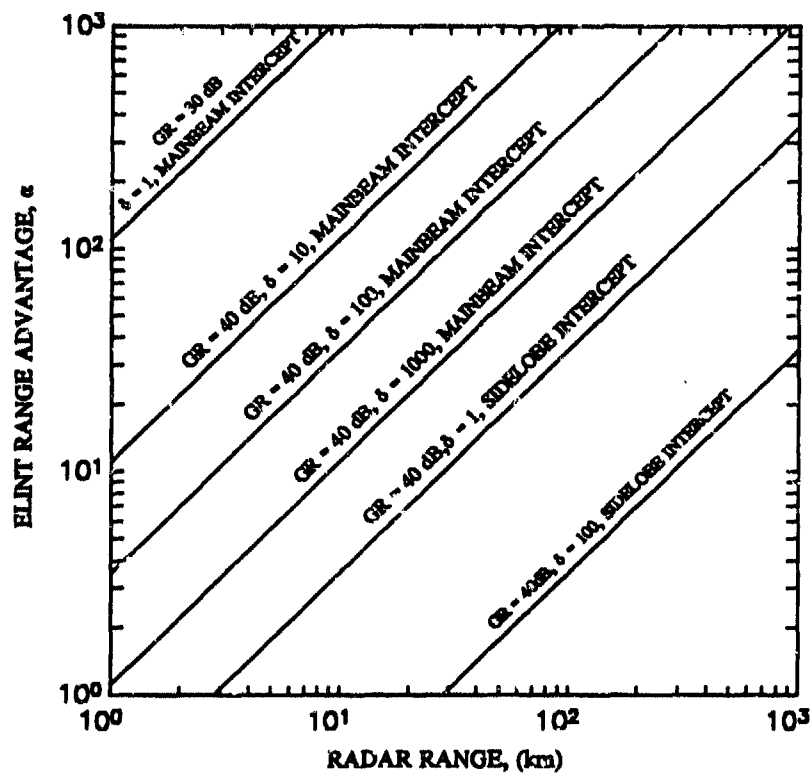


Figure 18. ELINT Range Advantage

APPENDIX B. MATLAB SOURCE CODE

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% WORLD.M
% Written by Gary Sipe
%
%
% This program is a menu-driven utility used to write the geometry files
% used by MAKETGT.M to construct target signal files. It is menu driven
% and its operation is self explanatory. The filename given when requested
% by the program must no more than 5 characters in length. The data
% output by WORLD will be stored in the file FNAME-G.MAT where
% "FNAME" is the filename you specified at the prompt.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;
clg;

% Initialize the default values of the system variables.

samp_freq=400000;      %in Hz
prf=9000;              %in Hz
pw=5e-6;               %in seconds
beam_w=7.5;            %in degrees
scan_rate=1;           %in sweeps/second
duration=0.05;         %in seconds

x1=-25;                %in km
y1=0;                  %in km

x2=25;                 %in km
y2=0;                  %in km

xt=50;                 %in km
yt=100;                %in km

v=[300 330];           %in [km/hr bearing]

menut='Setup Options';

```

```

menu1='Change Radar Parameters';
menu2='Change Receiver Parameters';
menu3='Change Target Parameters';
menu4='Load Old File';
menu5='All Done';

done=0;
format compact

% Enter the main program loop. The loop does nothing more than display current
% settings and prompt the user to make changes. Resulting changes may be
% saved to a file or discarded.

while done==0,
    clc;
    disp(['PRF = ' int2str(prf) ' Hz']);
    disp(['Pulse Width = ' int2str(pw*1e6) ' microseconds']);
    disp(['Beam Width = ' num2str(beam_w) ' degrees']);
    disp(['Scan Rate = ' num2str(scan_rate) ' sweeps/second']);
    disp(['Duration of Problem = ' num2str(duration) ' seconds']);
    disp("");
    disp('Receiver and Target Coordinates are in kilometers');
    disp("");
    disp(['Sample Frequency = ' int2str(samp_freq) ' Hz']);
    disp(['Receiver 1 initial position = (' num2str(x1) ',' num2str(y1) ')']);
    disp(['Receiver 2 initial position = (' num2str(x2) ',' num2str(y2) ')']);
    disp(['Receiver velocity = ' num2str(v(1)) ' km/hr']);
    disp(['Receiver direction = ' num2str(v(2)) ' degrees true']);
    disp(['Target initial position = (' num2str(xt) ',' num2str(yt) ')']);

    choice=menu(menu,menu1,menu2,menu3,menu4,menu5);
    if choice==1,
        clc;
        new=input(['Input the New PRF (<ENTER>=' int2str(prf) ') ']);
        if ~isempty(new),
            prf=new;
        end;
        clear new
        new=input(['Input the New Pulse Width (<ENTER>=' int2str(pw*1e6) ') ']);
        if ~isempty(new),
            pw=new/1e6;
        end;
        clear new
        new=input(['Input the New Beam Width (<ENTER>=' num2str(beam_w) ') ']);

```

```

if ~isempty(new),
    beam_w=new;
end;
clear new
new=input(['Input the New Scan Rate (<ENTER>=' num2str(scan_rate) ') ']);
if ~isempty(new),
    scan_rate=new;
end;
clear new
new=input(['Input the New Duration of Problem (<ENTER>=' num2str(duration) ')
']);
if ~isempty(new),
    duration=new;
end;
clear new
elseif choice==2,
    clc;
    new=input(['Input the New Sample Freq (<ENTER>=' int2str(samp_freq) ') ']);
    if ~isempty(new),
        samp_freq=new;
    end;
    clear new
    new=input(['Input the x1 (<ENTER>=' num2str(x1) ') ']);
    if ~isempty(new),
        x1=new;
    end;
    clear new
    new=input(['Input the y1 (<ENTER>=' num2str(y1) ') ']);
    if ~isempty(new),
        y1=new;
    end;
    clear new
    new=input(['Input the x2 (<ENTER>=' num2str(x2) ') ']);
    if ~isempty(new),
        x2=new;
    end;
    clear new
    new=input(['Input the y2 (<ENTER>=' num2str(y2) ') ']);
    if ~isempty(new),
        y2=new;
    end;
    clear new
    new=input(['Input the New Velocity (<ENTER>=' num2str(v(1)) ') ']);
    if ~isempty(new),

```

```

        v(1)=new;
    end;
    clear new
    new=input(['Input the New Direction (<ENTER>=' num2str(v(2)) ' ' ']);
    if ~isempty(new),
        v(2)=new;
    end;
    clear new
elseif choice==3,
    clc;
    new=input(['Input the xt (<ENTER>=' num2str(xt) ' ' ']);
    if ~isempty(new),
        xt=new;
    end;
    clear new
    new=input(['Input the yt (<ENTER>=' num2str(yt) ' ' ']);
    if ~isempty(new),
        yt=new;
    end;
    clear new
elseif choice==4,
    fname_in=input('Input the problem name (5 chars/no extension) ','s');
    fname_in=[fname_in '-g'];
    eval(['load ' fname_in]);
elseif choice==5,
    done=1;
end;
end;

done=0;
ans=input('Save Changes to a Problem File? (y/n) ','s');
if ans=='y',
    while done==0,
        fname_out=input('Input the problem name (5 chars/no extension) ','s');
        fname_out=[fname_out '-g'];
        old=exist([fname_out '.mat']);
        if old==2,
            ans=input('File exists -- delete it? (y/n) ','s');
            if ans=='y',
                clear menu1 menu2 menu3 menu4 menu5 done ans choice menut
                eval(['save ' fname_out]);
                done=1;
            end;
        else,

```

```
clear menu1 menu2 menu3 menu4 menu5 done ans choice menu1
eval(['save ' fname_out]);
done=1;
end;
end;
end;
format loose
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% MAKETGT.M
% Written by Gary Sipe
%
% This program reads the geometry and radar data provided by the '-G' files %
% created with WORLD.M. Filenames are again only 5 letters in length. %
% The signal file created by MAKETGT will be saved in FNAME-S.MAT. %
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% Reset the environment.

```

```

clear
clc
rand('normal');

```

```

% Get the problem file of interest.

```

```

clc;
fname=input('Input the problem name (5 chars/no extension) ','s');
eval(['load ' fname '-g']);

```

```

ls=2.997925e5;          %(kilometers/sec)
total_l=duration*samp_freq;
delay=5000;
toa_al=delay;

```

```

% Prompt regarding noise addition and initialize the data vectors accordingly.

```

```

ans=input('Do you want to add noise? ','s');
if ans=='y',
    data1=rand(1,total_l);
else,
    data1=zeros(1,total_l);
end;
data2=data1;
d=length(data1);

```

```

disp('Working');

```

```

% Calculate and build a burst envelope which will then be displaced in time in the
% vectors DATA1 and DATA2 to build the final data vectors.

```

```

nextscan=floor(samp_freq/scan_rate);
nexttime=nextscan/samp_freq;
numscans=ceil((total_l-delay)/nextscan);
target_l=floor((beam_w/360/scan_rate)*samp_freq);

```

```

envelope=sin(linspace(0,pi,target_l));
time=linspace(0,beam_w/360/scan_rate,target_l);
mod=(square(2*pi*prf*time,pw*100*prf)+ones(time)).*0.5;
target=envelope.*mod.*10;

% Build data vectors using the burst envelope created above.
for c=1:numscans,
    xd=sqrt((x2(c)-xt)^2+(y2(c)-yt)^2)-sqrt((x1(c)-xt)^2+(y1(c)-yt)^2);
    xdn=round(xd*samp_freq/l);
    if xdn+target_l+toa_al(c)<=d,

data1(1,toa_al(c):toa_al(c)+target_l-1)=data1(1,toa_al(c):toa_al(c)+target_l-1)+target;
    toa_a2(c)=toa_al(c)+xdn;

data2(1,toa_a2(c):toa_a2(c)+target_l-1)=data2(1,toa_a2(c):toa_a2(c)+target_l-1)+target;
    toa_al(c+1)=toa_al(c)+nextscan;
    end;
    if c~=numscans,
        x1(c+1)=x1(c)+v(1)*nexttime*sin(v(2)*pi/180)/3600;
        x2(c+1)=x2(c)+v(1)*nexttime*sin(v(2)*pi/180)/3600;
        y1(c+1)=y1(c)+v(1)*nexttime*cos(v(2)*pi/180)/3600;
        y2(c+1)=y1(c+1);
    end;
end;

clc;

% Prompt user for filename and save the necessary data.
% Extraneous variables are cleared prior to saving.
ans=input('Do you want to save this target? (y/n) ','s');
if ans == 'y',
    clear c ans mod target target_l target_t toa_a toa_a2(c) nexttime
    clear numscans con ls d delay xdn total_l duration
    clear time nextscan envelope
    eval(['save ' fname '-s']);
end;

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% MAIN.M
% Written by Gary Sipe
%
% This program is the central processing loop of the algorithm. It calls the
% subroutine PROCESS to perform the processing of individual pulse bursts.
% it will retrieve a signal file ('fname-s.mat') and output all parameters to an
% analysis file ('fname-a.mat').
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% Reset the environment.

```

```

clear
clg
format compact

```

```

% Retrieve the problem file of interest.

```

```

clc;
fname=input('Input the problem name. ','s');
loadstr=['load ' fname '-s'];
eval(loadstr);
time=linspace(0,length(data1)/samp_freq,length(data1));
new=[];

```

```

% The next series of program blocks prompts the user for various options.

```

```

ans=input('Do you want to view the target? (y/n) ','s');
if ans == 'y',
    plot(time,data1),grid,xlabel('Time -- seconds'),ylabel('Magnitude'),
    title(['Plot of Data for Target File -- ' fname])
    pause;
end;

```

```

ans=input('Do you want to view the intermediate results? (y/n) ','s');
if ans == 'y',
    show=1;
else,
    show=0;
end;

```

```

ans=input('Do you want keyboard access between results? (y/n) ','s');
if ans == 'y',
    key=1;
else,
    key=0;

```

```

end;

clc;
disp('Beginning Analysis --');
disp('');

[N_mean,N_var,N_first,N_totpwr]=myfft(data1(1:1024),samp_freq);
[S_mean,S_var,S_first,S_totpwr]=myfft(data1(5001:13192),samp_freq);

a=3;
pri=S_first;
if N_totpwr ~=0,
    SNR=10*log10(S_totpwr/8/N_totpwr);
    thresh=N_mean+a*N_var;
else,
    thresh=1;
    SNR='undefined';
end;
disp(['Auto threshold is -- ' num2str(thresh) ' <ENTER> to accept']);
new=input('Or type in your own threshold now');
if ~isempty(new),
    thresh=new;
end;

for sens=1:2;
    if sens==1,
        disp('Receiver 1 Simulation');
        data=data1;
    else,
        disp('Receiver 2 Simulation');
        data=data2;
        pri=S_first;
    end;

    disp('In Loop 1');
    for c1=1:length(data),
        if data(c1)>thresh,
            if data(c1+1)>thresh,
                disp('detect');
                scan=1;
                [pri,xhat,start,done]=process(data,c1,samp_freq,pri,thresh,show,key);
                [toa(scan)]=centroid(xhat,samp_freq);
                break;
            end;
        end;
    end;
end;

```

```

    end;
end;

disp('in Loop 2')
done=done+1000;
lastscan=start;
for c2=done:1:length(data),
    if data(c2)>thresh,
        if data(c2+1)>thresh,
            disp('detect');
            start=c2;
            scan=2;
            [pri,xhat,start,done]=process(data,c2,samp_freq,pri,thresh,show,key);
            [toa(scan)]=centroid(xhat,samp_freq);
            scan_rate(1)=samp_freq/(start-lastscan);
            lastscan=start;
            break;
        end;
    end;
end;

c3=done+1000;
finished=0;
disp('In Loop 3')

while finished==0,

    if c3>=length(data);
        break;
    else;
        c3=c3+1;
    end;

    if data(c3)>thresh,
        if data(c3+1)>thresh,
            disp('detect');
            [pri,xhat,start,done]=process(data,c3,samp_freq,pri,thresh,show,key);
            scan_obs=samp_freq/(start-lastscan);
            scan_rate(scan)=scan_rate(scan-1)+(scan_obs-scan_rate(scan-1))/scan;
            scan=scan+1;
            lastscan=start;
            [toa(scan)]=centroid(xhat,samp_freq);
            c3=done+1000;

```

```

        end;
    end;
end;

if sens==1,
    toa1=toa;
    pri1=pri;
    scan_rate1=scan_rate;
    scan_rate=0;
else,
    toa2=toa;
    pri2=pri;
    scan_rate2=scan_rate;
end;

end;

clear data data1 data2 time
eval(['save ' fname '-a']);
format loose

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% MYFFT.M
% Written by Gary Sipe
%
% This function is called by MAIN to obtain spectrum information during
% the first detection of an emitter. It produces the average value, the
% variance, the first spectral line, and the total power.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function [avg,var,first,totpwr]=myfft(data,samp_freq);
%function [avg,var,first,totpwr]=myfft(data,samp_freq);

```

```

l=length(data);
y=fft(data);
avg=y(1)/l;
spec_den=(y.*conj(y))/l;
totpwr=sum(spec_den);
var=cov(data);
delf=400000/l;
[dummy,first]=max(y(2:250));
first=(first-1)*delf;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% PROCESS.M
% Written by Gary Sipe
%
% This function is called by MAIN to handle burst processing. It returns the
% estimates of pri, the burst envelope, the burst start time, and the burst
% ending time. This function also handles all truncation logic.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function [pri,xhat,start,done]=process(data,start,samp_f,pri,thresh,show,key);
%function [pri,xhat,start,done]=process(data,start,samp_f,pri,thresh,show,key);

```

```

% The variables 'show' and 'key' are flags used to allow displays during the processing
% cycle.

```

```

if nargin==5,
    show=0;
else
    show=show;
end;

```

```

if nargin==6,
    key=0;
else,
    key=key;
end;

```

```

% Set all flags and initial values.

```

```

tau=1/samp_f;
broken=0;
xhatold=0;
trunc=12;
cold=zeros(1,trunc);
a=0;
p=length(pri);
k=0;
pri_old=pri(p);
nuls=0;
disp('In process');

```

```

for c=start:1:length(data),
    nuls=nuls+1;
    if data(c)>thresh,
        if data(c+1)>thresh,

```

```

nuls=0;
if cold(trunc)~=0,
    % Truncate the first few pulses based on the 'trunc' setting.
    if a<=trunc,
        a=a+1;
    end;
    if a==trunc,
        a=a+1;
        start=c;
    end;
    % Beginning of burst processing routine.
    if a>trunc,
        % PRI logic and filter.
        p=p+1;
        pri_hat=(samp_f/(c-cold(trunc)));
        pri(p)=pri_old+(pri_hat-pri_old)/(p+1);
        pri_old=pri(p);
        % Envelope logic and filter.
        k=k+1;
        xhat(k)=xhatold+(1/k)*((data(c)+data(c+1))-xhatold);
        xhatold=xhat(k);
    end;
end;
% Simulate a set of shift registers to be used when truncating the end of the burst.
cold=shiftl(cold,c);
end;
end;
% This is the end of burst logic.
if a>trunc,
    if nuls>trunc*samp_f/pri(p);
        done=cold(1);
        pri=pri(1:(length(pri)-trunc));
        xhat=xhat(1:(length(xhat)-trunc));
        xhat(length(xhat)+1)=start;
        xhat(length(xhat)+1)=cold(1);
        broken=1;
        break;
    end;
end;
end;
end;

```

```

% Exit logic.
if broken==0,
    done=length(data);
    pri=pri(1:(length(pri)-trunc));
    xhat=xhat(1:(length(xhat)-trunc));
    xhat(length(xhat)+1)=start;
    xhat(length(xhat)+1)=cold(1);
end;

% Display logic.
if show~=0,
    subplot(211),plot(pri),grid
    subplot(212),plot(xhat(1:length(xhat)-2)),grid
    pause;
    clg;
end;

% Keyboard utility logic.
if key==1,
    keyboard;
end;

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% CENTROID.M
% Written by Gary Sipe
%
% This function is called by MAIN to numerically integrate the estimated
% burst envelope.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function [toa]=centroid(xhat,samp_freq);
%function [toa]=centroid(xhat);
l=length(xhat);
minx=xhat(l-1);
maxx=xhat(l);
xhat=xhat(1:l-2);
M=0;
Mx=0;
for count=1:l-3,
    M=M+0.5*(xhat(count)+xhat(count+1));
    Mx=Mx+0.5*(count+0.5)*(xhat(count)+xhat(count+1));
end;
x_bar=Mx/M;
toa=minx+x_bar*(maxx-minx)/(l-2);
toa=toa/samp_freq;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                    %
%  SHIFT.M                                                            %
%  Written by Gary Sipe                                              %
%                                                                    %
%  Simulates a block of shift registers.                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function [data]=shift(data,new);
l=length(data);
data(1:l-1)=data(2:l);
data(l)=new;
return;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% BIRDSEYE.M
% Written by Gary Sipe
%
% This program file is used to provide the graphical displays of estimated
% lines of position over time. It reads the analysis files (fname-a.mat).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clg

ls=2.997925e5; % Speed of light (km/sec)

fname=input('Input the problem file name ','s');
eval(['load ' fname '-a']);

axis('square')
axis([-50 150 0 350]);
plot(xt,yt,'o')
hold on;
xoff=0;
yoff=0;
for c=1:length(toa1),
    dt=toa1(c)-toa2(c);
    dtmax=sqrt((x1(c)-x2(c))^2+(y1(c)-y2(c))^2)/ls;
    cnt=sqrt((x1(c)-x2(c))^2+(y1(c)-y2(c))^2)/2;
    a=cnt*dt/dtmax;
    b=sqrt(cnt^2-a^2);
    xoff=(x1(c)+x2(c))/2;
    yoff=(y1(c)+y2(c))/2;
    x=linspace(0,500,1000);
    LOP=(x)*b/a+yoff;
    plot(x1(c),y1(c),'rx'),grid
    plot(x2(c),y2(c),'rx'),grid
    plot(x+xoff,LOP,'g-')
    pause(3);
end;
xlabel('X in km. ');
ylabel('Y in km. ');
hold off;
axis('normal');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% EXAMINE.M
% Written by Gary Sipe
%
% This utility is used to interactively display and save the contents of very
% variables. Screen buttons are drawn on the MATLAB display and the
% contents can be scrolled. Many other options are also available such as
% plot labels, changing axes, changing starting point and block sizes of the
% display, and saving the current screen to a meta file. This is a utility
% function used during debugging and no documentation is provided.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
function examine(time,data);
```

```

clg
clc

```

```

if nargin==1,
    data=time;
    time=linspace(1,length(data),length(data));
end;

```

```

maxval=max(data);
minval=min(data);
l=length(data);
disply=1;
hardcopy=0;
getname=1;
reverse=0;
forward=1;

```

```

tstring="";
xstring="";
ystring="";

```

```

menutitle='Labels Menus';
menu1='Change Title';
menu2='Change Xlabel';
menu3='Change Ylabel';
menu4='Place a Text Marker';
menu5='Preview Your Plot';
menu6='Save Plot to a Meta File';
menu7='Back to Plot';

```

```

clc
ans=input('Will you want to save plots to a meta file? (y/n) ','s');
if ans=='n',
    nomet=1;
else,
    while getname==1;
        nomet=0;
        clc
        disp('Input the filename for meta files (8 chars no extension) ');
        metaname=input('8 chars no extension (n=none) ','s');
        metaname=[metaname '.met'];
        ans=exist(metaname);
        if ans==2;
            ans=input('File exists -- delete it? (y/n) ','s');
            if ans=='y'
                eval(['!del ' metaname]);
                getname=0;
                eval(['meta ' metaname]);
            end;
        else,
            getname=0;
            eval(['meta ' metaname]);
        end;
    end;
end;

a=1;
block=100;

butdx=[0.15 0.215 0.215 0.15 0.15];
butlx=[0.26 0.35 0.35 0.26 0.26];
butax=[0.35 0.44 0.44 0.35 0.35];
buthcx=[0.44 0.53 0.53 0.44 0.44];
butsx=[0.575 0.665 0.665 0.575 0.575];
butbx=[0.665 0.755 0.755 0.665 0.665];
butrx=[0.80 0.85 0.85 0.80 0.80];
butfx=[0.85 0.90 0.90 0.85 0.85];
buty=[0.938 0.938 0.888 0.888 .938];
axis([time(a) time(a+block) minval maxval])

```

% The remainder of the program is the main loop. Tracing the operation of this loop
% and the logic required for the screen buttons is not recommended.

```
while disply==1;
    if hardcopy==0,
        axis([time(a) time(a+block) minval maxval])
        displot(time(a:a+block),data(a:a+block)),grid
        title(tstring),xlabel(xstring),ylabel(ystring);
        polyline(butdx,puty,'sc')
        text(0.155,0.898,'Done','sc')
        polyline(butlx,puty,'sc')
        text(0.265,0.898,'Labels','sc')
        polyline(butax,puty,'sc')
        text(0.37,0.898,'Axis','sc')
        if nomet==0,
            polyline(buthcx,puty,'sc')
            text(0.46,0.898,'Meta','sc')
        end;
        polyline(butsx,puty,'sc')
        text(0.585,0.898,'Start','sc')
        polyline(butbx,puty,'sc')
        text(0.675,0.898,'Block','sc')
        if reverse==1,
            polyline(butrx,puty,'sc')
            text(0.805,0.898,'Rev','sc')
        end;
        if forward==1,
            polyline(butfx,puty,'sc')
            text(0.855,0.898,'Fwd','sc')
        end;

        [x,y]=ginput(1,'sc');
        axis;
```

```
if x>0.80&x<0.85&y>0.888&y<0.938,
    if reverse==1,
        if a-block<=1;
            a=1;
            reverse=0;
            forward=1;
        else;
            a=a-block;
            reverse=1;
            forward=1;
        end;
    end;
```

```

    end;
end;
elseif x>0.85&x<0.90&y>0.888&y<0.938,
    if forward==1,
        if a+block>=l-block,
            a=l-block;
            reverse=1;
            forward=0;
        else,
            a=a+block;
            reverse=1;
            forward=1;
        end;
    end;
elseif x>0.15&x<0.215&y>0.888&y<0.938,
    disp=0;
elseif x>0.26&x<0.35&y>0.888&y<0.938,
    loop=1;
    while loop==1;
        clc
        disp(['Current title is: ' tstring])
        disp(['Current Xlabel is: ' xstring])
        disp(['Current Ylabel is: ' ystring])
        disp("")
        disp("")
        choice=menu(menuitle,menu1,menu2,menu3,menu4,menu5,menu6,menu7);
        if choice==1,
            tstring=input('Input the new title ','s');
        elseif choice==2,
            xstring=input('Input the new Xlabel ','s');
        elseif choice==3,
            ystring=input('Input the new Ylabel ','s');
        elseif choice==4,
            tmstring=input('Input the string for the text marker ','s');
            axis([time(a) time(a+block) 0 maxval]);
            dispot(time(a:a+block),data(a:a+block)),grid
            title(tstring),ylabel(ystring),xlabel(xstring)
            gtext(tmstring),
            pause(2);
            axis;
        elseif choice==5,
            axis([time(a) time(a+block) 0 maxval]);
            dispot(time(a:a+block),data(a:a+block)),grid
            title(tstring),ylabel(ystring),xlabel(xstring)

```

```

        pause;
        axis;
    elseif choice==6,
        meta;
    elseif choice==7,
        loop=0;
    end;
end;
elseif x>0.35&x<0.44&y>0.888&y<0.938,
    clc
    disp('Values for the x-axis are scaled by the plot window ')
    disp('based on the starting point and block size. You may only ')
    disp('scale the y-axis of the plot.')
    disp("")
    disp("")
    new=[];
    new=input(['Input the new y-min (<ENTER>=' num2str(minval) ' ') ']);
    if ~isempty(new),
        minval=new;
    end;
    new=[];
    new=input(['Input the new y-max (<ENTER>=' num2str(maxval) ' ') ']);
    if ~isempty(new),
        maxval=new;
    end;
elseif x>0.575&x<0.665&y>0.888&y<0.938,
    new=[];
    clc
    new=input(['Input the new starting point (<ENTER>=' int2str(a) ' ') ']);
    if ~isempty(new),
        a=new;
        reverse=1;
        forward=1;
    end;
    if a<=1,
        a=1;
        reverse=0;
        forward=1;
    elseif a+block>=l-block,
        a=l-block;
        reverse=1;
        forward=0;
    end;
elseif x>0.665&x<0.755&y>0.888&y<0.938,

```



```

new=[];
clc
new=input(['Input the new block size (<ENTER>=' int2str(block) ' ) ']);
if ~isempty(new),
    if new>=1,
        block=l-1;
        a=1;
        forward=0;
        reverse=0;
    elseif new<1,
        block=1;
    else,
        block=new;
    end
end;
if a+block>=l-block,
    a=l-block;
    reverse=1;
    forward=0;
end;
elseif nomet==0,
    if x>0.44&x<0.53&y<9.38&y>0.888,
        if hardcopy == 1;
            hardcopy = 0;
        else;
            hardcopy=1;
        end;
    end;
end;
elseif hardcopy==1,
    displot(time(a:a+block),data(a:a+block)),grid
    title(tstring),ylabel(ystring),xlabel(xstring)
    meta;
    text(0.40,0.7,'Press Any Key to Continue','sc')
    pause;
    hardcopy=0;
    axis;
end;
end;
disp("");
clc;

```

LIST OF REFERENCES

1. Wiley, R. G., *Electronic Intelligence: the Analysis of Radar Signals*, Syracuse Research Corporation, 1982.
2. Wiley, R. G., *Electronic Intelligence: The Interception of Radar Signals*, Artech House, Inc., 1985.
3. Finney, R. L. and Thomas, G. B., *Calculus*, Addison-Wesley Publishing Company, 1989.
4. Mika, Frank J., "Fast Envelope Correlation for Passive Ranging," Master's Thesis, Naval Postgraduate School, Monterey, California, March 1991.
5. Graupe, D., *Time Series Analysis, Identification, and Adaptive Filtering*, Robert E. Krieger Publishing Company, 1984.
6. Gelb, A., and others, *Applied Optimal Estimation*, M.I.T Press, 1974.
7. Strum, R. D., and Kirk, D. E., *Discrete Systems and Digital Signal Processing*, Addison-Wesley Publishing Company, 1989.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3. Chairman, Code EC Department of Electrical & Computer Engineering Naval Postgraduate School Monterey, CA 93943-5002	1
4. Professor Harold Titus, Code EC/Ts Department of Electrical & Computer Engineering Naval Postgraduate School Monterey, CA 93943-5002	2
5. Professor Ralph Hippenstiel, Code EC/Hi Department of Electrical & Computer Engineering Naval Postgraduate School Monterey, CA 93943-5002	1
6. Naval Research Laboratory Code 9120 4555 Overlook Ave. Washington, DC 20375-5320	1
7. LT Gary Sipe 50904 Hollyhock Rd. South Bend, IN 46637	1